RELAY SELECTION IN BLUETOOTH MESH NETWORKS

Evaluation of a Distributed Self-Configuration Algorithm

A project by: Jacob Kjærsgaard, Martin Eriksen, Rasmus Liborius Bruun and Stamen Siromahov
Network and Distributed Systems 8. Semester, Aalborg University, Denmark

AALBORG UNIVERSITY
STUDENT REPORT

AALBORG UNIVERSITY
STUDENT REPORT

Institute of Electronic Systems
Fredrik Bajers Vej 7
DK-9220 Aalborg Ø

Title:
Relay Selection in Bluetooth Mesh Networks

Theme:
Networks and distributed systems

Project Period:
8. Semester, spring 2017

Project Group:
17GR820

Participants:
Jacob Kjærsgaard,
Martin Eriksen,
Rasmus Liborius Bruun and
Stamen Siromahov.

Supervisors:
Tatiana Kozlova Madsen
Rasmus Abildgren

Copies: 7

Page Numbers: 74

Date of Completion:
May 30, 2017

Abstract:

This project designs a distributed self-configuration algorithm for relay selection in ad-hoc networks. The algorithm is verified and evaluated using simulations. For the simulations, Bluetooth mesh has been used as the ad-hoc network. Bluetooth mesh is a new standard on top of Bluetooth low energy. This project gives insights into Bluetooth mesh and the constraints it adds to a self-configuration algorithm.

The focus of the algorithm was to provide a minimum relay selection, but during investigation it was found the minimum relay selection was not always desirable, as it gives a higher latency and hop distance in the network. The algorithm is evaluated on its relay selection and resources used.

It was possible to design a distributed self-configuration algorithm. The evaluation of the algorithm showed that it always obtained a relay selection giving full connectivity of the network. Denser networks cause the algorithm to select more relays which also results in higher collision probability.

Table of Contents

Preface

This report is created by Jacob Kjærsgaard, Martin Eriksen, Rasmus Liborius Bruun and Stamen Siromahov. The project is made on 8. semester Network and Distributed Systems and has been carried out in the spring of 2017 at Aalborg University(AAU). This project addresses the design of a self-configuring algorithm for a Bluetooth mesh network.

This project has been supervised by Tatiana Kozlova Madsen (associated professor at AAU), and Rasmus Abildgren (PhD and engineer at Samsung Denmark, Research Center). The authors would like to thank Samsung for the resources provided for the project.

The reader is expected to have fundamental knowledge corresponding to an 8th-semester student at Network and Distributed Systems, Aalborg University. However, prior knowledge of Bluetooth and Bluetooth mesh is not assumed.

The project involves network simulation in the simulator ns-3. Implementation of BLE and BLE-mesh in ns-3 has been developed during the project as a cooperation between Samsung Cambridge Solution Centre Ltd, a 6th computer science group, d602f17, and ourselves. The main developer of the BLE core is Clive D.W. Feather (Principal Engineer at Samsung Cambridge Solution Centre). The members of d602f17 are: Jonas Kloster Jacobsen, Magnus Nørhave Hansen, Marius Lauge Nørgaard and Mark Holst.

Sources are referenced [n] according to IEEE citation reference standard, where n is a number represented in the bibliography, which is placed at the end of the report. The primary appendices are placed after the bibliography and the secondary appendices such as simulator code (ns-3) and MATLAB scripts can be found online connected to the submission of this project in digital exam. All confidence intervals presented are 95% unless otherwise stated.

Computer simulation is the main evaluation method used in the project. Computer simulation is chosen over analytical evaluation, because analytical modeling of the entire BLE and BLE mesh stack behavior is infeasible. Test bed implementation would be feasible, but a test bed is a restricting factor when it comes evaluation. Therefore, computer simulation is the evaluation method of choice.

1 | Introduction

The internet made it possible for people to connect in ways we never imagined. The internet of things (IoT) gives physical devices the opportunity to connect and enrich many aspects of our daily lives. IoT is the sole enabler of the "smartification", which will soon come to application areas like homes, transports, health, buildings and cities. Omnipresence of smart, connected devices optimizes the way we use and produce resources, and they will be the foundation of revolutionary new services and applications, that are beyond our current imagination. The IoT revolution is imminent, and it is economically desirable to be part of the revolution. Market research institutes like Gartner and McKinsey estimate the IoT market to reach a several trillion-dollar worth within the next 10 years [1, 2].

In 2015 Samsung, the manufacturer of a multitude of consumer devices and domestic appliances, announced a grand vision for the IoT. By 2020 every product shipped by the company will be IoT enabled [3]. Every television set, toaster, Heating, ventilation and air conditioning (HVAC) and lightbulb will be equipped with electronics enabling device to device communication. It is no easy task to design a network technology, that works well for both coffee machines and laptop computers, and it is not a task for a single company. Instead, Samsung decided that all their devices should be built on an open ecosystem [3], with standardized networking technologies.

IoT capable network technologies are still in their early stages, and more are being developed. However, development is not without challenges. Besides being low cost, secure and suitable for resource constrained devices, networking technologies should work without special networking equipment, require minimal human involvement, handle large amounts of connected devices and cope with devices that move around. One such technology is being prepared by the Bluetooth Special Interest Group (SIG). They are designing an extension to the Bluetooth Low Energy (BLE) technology, that enables mesh networking on existing BLE chips. The Bluetooth mesh specification addresses many of the beforementioned challenges for IoT networks, however, some are still unsolved. Particularly, the Bluetooth mesh specification does not rule out human involvement in setup and maintenance of the network.

This project investigate how configuration of Bluetooth mesh networks can become autonomous. A distributed self-configuration algorithm is proposed. Validation and evaluation on the algorithm is carried out using computer simulation.

2 | Preliminary analysis

Every standards organization is preparing for the rise of the internet of things (IoT). The Bluetooth Special Interest Group is releasing a specification, which adds mesh capabilities to the Bluetooth standard. This chapter serves as an introduction to the IoT, Bluetooth and Bluetooth mesh. Firstly, the term IoT is explained. Secondly, the basics of Bluetooth are presented followed by an introduction to the functionality and limitations of the Bluetooth mesh. Further examination of self-configurability in mesh networks follows, and from this a problem statement is derived, which serves as the basis of the work carried out in this project.

2.1 IoT

More and more smart products are emerging. Our thermostats are no longer controlled by physical dials. Colorful lightbulbs can be controlled from our smartphones. Intrusion systems can determine when a window is broken or a door is opened. Soon, devices like these and many others will be ubiquitous. Networks with many nodes, with low resource consumption, exchanging small pieces of information arise. The networks providing communication for smart devices combined with the devices themselves are called the internet of things (IoT).

IoT is a hot topic and the term is widely (over)used especially in economic predictions and marketing. There is no clear definition of the term and it seems that every book, research report and article addressing the topic has their own definition. Some definitions are technical and emphasize the 'Internet' part of the term, e.g. in the book *6LoWPAN: The Wireless Embedded Internet,* by Zach Shelby, Carsten Bormanny, who states that the term IoT

"Encompasses all the embedded devices and networks that are natively IP-enabled and Internet-connected, along with the Internet services monitoring and controlling those devices." [4]

Other definitions are broad and more business relevant, e.g. in the book *Internet of Things: Legal Perspectives*, by Rolf H. Weber, Romana Weber, who use the following definition:

"a world where physical objects are seamlessly integrated into the information network, and where the physical objects can become active participants in business processes. Services are available to interact with these 'smart objects' over the Internet, query their state and any information associated with them, taking into account security and privacy issues" [4]

Nick Wainwright (HP Labs and Chair of the UK Future Internet Strategy Group) used a simplistic description in his *Innovate 11 Presentation - Internet Of Things Panel*, where he simply stated:

"Services + Data + Networks + Sensors = Internet of Things" [4]

The term is broad and can be tweaked to focus on things like security, privacy, business models, services, network topology, interoperability/compatibility, computation, etc. More examples of definitions can be found in [4].

IoT has immense potential to positively impact efficiency, safety and convenience and has a vast amount of application areas. Application areas include:

- Sensing of pollution and sewer levels in cities, which could be used to warn citizens and dispatch authorities information about transport (cars in a city, containers being shipped around the world, etc.) could be used to intelligently route and coordinate to reduce transport time.
- Control of heating, lighting, air quality and door-locks in an office building could be intelligently and remotely controlled to reduce utility costs and improve security

These are only a few examples of possible applications of IoT. While these have potential to improve on existing processes, there are still other unthought of applications applicable to processes and services that do not exist today. Generally, applications of IoT include a vast number of devices. Even though these devices will eventually be connected to the internet, and altogether have a large footprint on the traffic patterns, each individual device will most likely only generate a small amount of network traffic.

IoT included somewhere between 6.4 billion and 17.6 billion devices in 2016 depending on which analysis institute you ask. [5] Growth projection also differ but most institutes expect around 30 billion devices by 2020. [5] These predictions make IoT one of the fastest growing markets, where a lot of capital is involved. Gartner estimates that consumers and businesses collectively will be spending around 3 trillion dollars on IoT devices and services in 2020. [6]

There are good reasons to get into IoT, and whether the motivation is profit, improving quality of life or saving the environment, it is inevitable that IoT devices will become ubiquitous. However, for IoT to really take off, standardization is essential. Having every company creates their own proprietary ecosystem is destructive. There is a need for standards on different levels. From a communication standpoint, standardized protocols should be few and wide spread. When smaller and smaller devices are being connected, there is a need for low power networking which covers a lot of ground. One such effort is the development of wireless mesh networks technologies.

2.2 Wireless mesh networks

A wireless mesh network is a network in which nodes contributes to the network by relaying messages. Specialized networking equipment is not necessary. Depending on the capabilities of the participating nodes, routing in the network may be done by simple flooding or a more complex routing protocol. Through this cooperation the network is able to cover a wider area compared to "normal infrastructure", where wireless devices can only communicate with a central access point responsible for relaying their packets, as seen in Figure 2-1. The blue area is the "normal infrastructure" coverage and the white area is for the mesh coverage.

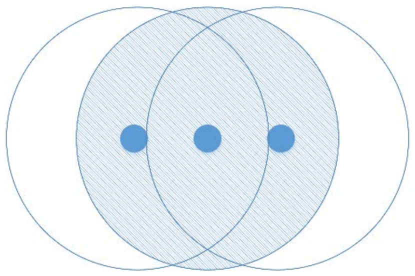

Figure 2-1 Area coverage, blue is covered by normal infrastructures and white is mesh coverage

Furthermore, the network is using its own nodes to set up the network, decreasing price and setup time compared to a normal network infrastructure were cabling or wireless access point must be installed and configured. The drawbacks of this type of network come from the fact that nodes act as both a router and an endpoint consuming more power to operate. The latency is getting increased and the bandwidth is shared and thereby decreases, due to the message having to do more hops, compared to a direct link. Wireless mesh network is a suitable solution for IoT networks transferring small messages, since it keeps the infrastructure price down, and makes area expansion easy. As mentioned, IoT devices are often expected to be low power and low cost. Therefore, it has been chosen to investigate the new and emerging technology Bluetooth mesh, which is a mesh stack implemented on top Bluetooth low energy, offering low power consumption and cheap cost. This is done by first considering Bluetooth and then Bluetooth low energy and from this understanding how the Bluetooth mesh stack is implement.

2.3 Bluetooth

Bluetooth is a data transfer wireless technology standard developed by Ericsson in 1994. It was designed as a low power consuming alternative to the serial cables. Today it is mainly used for connections between a wireless accessory and a PC/smartphone. Bluetooth is well suited for connecting battery driven devices that continuously stream data over short distances due to its low power consumption and low cost compared to Wi-Fi.

Bluetooth is most commonly referred to as a short distance data transfer technology. This is a common misconception since Bluetooth under optimal conditions can reach distances of up to 100m within line-of-sight. The maximum transfer distance is determined by the power class of the devices communicating. There are 3 power classes, class 1 allowing up to 100m range, class 2 up to 10m and class 3 up to 1m range. Due to class 3's very limited range and class 1's very long range which gives a high-power consumption these two classes are rarely used. Due to this class 2 is the most commonly used class. When devices with different power classes communicate, the range is determined by the shorter ranged device.

Bluetooth devices operate unlicensed in the 2.4 GHz ISM band range. This range of frequencies is shared with other networking technologies like IEEEs 802.11 (used in Wi-Fi) and 802.15.4 (used in e.g. ZigBee).

Originally, Bluetooth was designed as a connection oriented technology. However, in version 4.0 of the specification, Bluetooth SIG introduced the connectionless Bluetooth Low Energy (BLE), which is quite different from the existing technology. Bluetooth is generally divided into two technologies: Bluetooth classic and BLE.

2.3.1 Bluetooth classic

Bluetooth classic can be divide into 3 different groups with different physical and mac layer protocol: Basic Rate(BR), Enhanced Data Rate(EDR) and Alternate MAC/PHY(AMP). BR and EDR phy/mac protocols operate on 79 channels with 1 MHz bandwidth. BR provide speeds up to 1 Mb/s and EDR provide speeds up to 3 Mb/s.

AMP is an extension to the Bluetooth standard enabling the usage of other MAC and physical layer protocols, e.g. the use of an 802.11 radio to provide Bluetooth services. The maximum speed of AMP is dictated by the specific protocols used.

Bluetooth is a packet-based protocol with a master-slave structure that allows for up to 7 devices to connect to 1 master device creating an ad hoc Bluetooth network also called a piconet. Within this network, packet exchange is based on a basic clock interval set up by the master and shared with all the connected devices. It is possible for devices in this piconet to be a master, while being a slave in another piconet, thus allowing multiple piconets to be connected into a scatter net.

2.3.2 Bluetooth Low Energy

Bluetooth Low Energy uses a completely new protocol stack to quickly build up simple links. Compared to classic Bluetooth, Low Energy uses only 40 channels with 2 MHz bandwidth and provides a maximum bit rate of 2 Mb/s, thought the default rate is 1 Mb/s. The new Bluetooth Low Energy protocol allows ultra-low power consumption.

Low implementation cost makes it a suitable choice of protocol to be used in a IoT wireless mesh network, especially because IoT devices are commonly low powered and produced in substantial numbers. This also means that a BLE device will most likely be limited on computation and memory.

Since Bluetooth BR/EDR and BLE were intended for different uses, it is possible to buy chips, which only implement one of the technologies. These chips are called single-mode, and allow for cheap manufacturing of one purpose devices, e.g. a Bluetooth headset or a keyring-tracker. Some devices (most commonly smartphones and computers) need the ability to communicate with both BR/EDR and BLE devices. For this purpose, dual-mode chipsets are available. BLE provides both point-to-point and broadcast mechanisms for transmitting messages. The broadcast capability is interesting in an IoT context. It can serve as the foundation of a wireless mesh network. This is exactly what Bluetooth mesh utilize.

2.4 Bluetooth mesh

The Bluetooth Special Interest Group has recognized the usefulness of bringing mesh networking capabilities to the Bluetooth low energy wireless technology. The mesh specification [7] allows Bluetooth devices to extend the otherwise limited range of the Bluetooth devices. This allows for messages to be carried to devices beyond the transmitting range of a single Bluetooth device. The Bluetooth mesh stack is built on top of Bluetooth Low Energy allowing devices that are part of the mesh network to relay messages. It consists of seven layers whose specification and functionality is briefly described in this section. Figure 2-2 depicts the mesh protocol stack. The blue layers are protocols specified by the Bluetooth mesh specification. The green layers are the layers interfacing to the Bluetooth mesh stack.

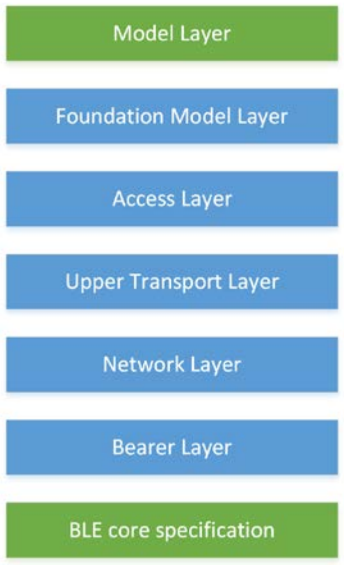

Figure 2-2 Bluetooth mesh stack

2.4.1 Model and Foundation Model Layers

The model and foundation model layers in the mesh stack is equivalent of the applications layer in the OSI model. Models are simply specific applications, which enable some functionality. The foundation models describe models required for configuration and management of the network. Foundation models include functionality for devices to provide information about which network roles it can take on, and handling of the setup of the network (includes management of security i.e. exchange of cryptographic keys). The foundation models are required by all mesh devices. Like the foundation model layer, the model layer defines applications which provide some functionality. The regular models are specific to the extra capabilities a node has. A light bulb for instance, would have some model describing how to interact with it, i.e. how to turn on and of the light bulb.

The model defines state, behavior and interaction, which is often client-server based. Taking the light bulb example, the light bulb would be a server, which provides the services to turn on and of the bulb. Interaction with the server would require the appropriate client, who has knowledge of how to interact with the server.

2.4.2 Access Layer

This layer creates the liaison between the upper transport layer and the models/applications in the higher layers. The access layer receives all messages from one lower layer, and needs to determine which exact model should receive each message. Besides distributing the messages, the access layer is responsible for ensuring that the messages are send by a valid source. Typically, the source would need to be part of the mesh network as well as provisioned to interact with the model. In the transmission of messages the access layer is responsible for managing the encryption method and key applied to the messages in the upper transport layer. This encryption provides confidentiality to the messages, so only the intended destination(s) are capable of understanding the message.

2.4.3 Upper Transport Layer

The upper transport layer performs authentication on messages based on the encryption specified by the access layer. Additionally, the upper transport layer is responsible for the integrity of the messages. This is done by adding an integrity check to the message. In this way, it can be detected if the message has been changed during transmission, and discarded if so.

If a node supports the friend role (explained in Section 2.5) the upper transport layer is responsible for managing the friendship. This includes saving messages addressed for the friend, and providing the messages to the friend upon request.

2.4.4 Lower Transport Layer

Bluetooth has some hard restrictions on the message sizes. To enable transfer of large messages, segmentation is required. The segmentation, if necessary, is handled by the lower transport layer. While unsegmented messages are send on a best effort basis, the segmented messages are sent reliably. Reliability is ensured by acknowledging segments, and retransmitting unacknowledged messages. However, when a message is multicast (has multiple receivers), reliability is not ensured on segmented messages. If reliability is desired on short messages, these can be send as segmented messages with one segment.

2.4.5 Network Layer

The network layer has the important function of relaying messages. Without any relays, the network would never exceed one hop. The relay functionality is turned on/off by the network administrator. The administrator should carefully choose which nodes should be relays, as to avoid overflow in the network while ensuring connectivity.

The network layer performs authentication on messages to ensure that is has been transmitted by a node which is part of the network. It does so by encrypting messages with a network key, which is known by every node on the network. Additionally, the destination address is scrambled, which obscures the origin of the message to any party who are not part of the network. The network layer avoids replaying the same message multiple times, by managing a sequence number for every source. Some messages are meant only for devices in close vicinity and should not propagate though the entire network. To enable this behavior every message has a time to live (TTL), which is decremented every time the message is relayed. Once the TTL is decremented to 1, the message is not relayed further.

2.4.6 Bearer Layer

The bearer layer handles the interface to the Bluetooth core. In the Bluetooth core, two physical links are available. One is a broadcasting link, which in Bluetooth terminology is called the advertising bearer. This is the bearer used in Bluetooth mesh, since is allows one node to send a message to multiple receivers in one transmission and without establishing a connection beforehand. This bearer has no collision avoidance and detection. The second bearer is point to point and based on an established connection. This bearer incorporates legacy devices into the mesh network, where a unit supporting both bearers can link a Bluetooth device which do not support mesh into the network.

2.4.7 Bluetooth mesh deployment scenarios

Mesh networks are advantageous for several scenarios, however, BLE-mesh is better suited for some scenarios than others, due to its low power consumption at the cost of a low data rate compared to other technologies such as Wi-Fi. BLE-mesh is perfect for smart buildings with a high density of electronic equipment would be perfect for short range BLE chips.

The vision is incorporating BLE-mesh capabilities in light bulbs, switches, coffee machines, printers, thermostats, alarms, ventilation, air conditioning and possibly many other devices. Use cases include convenient control of lighting, alarm systems and automation of heating systems. A huge advantage of having a mesh network in a building to which you can connect new devices is that equipping the building with new smart tech does not involve installation of new infrastructure. Running cables throughout a building or setting up wireless access points represent a significant investment, which can be evaded using mesh technology. Hereby, vendors can offer smart tech, which not only supplies a useful service/application but also is hassle-free to install and setup.

Naturally, the variety of applications/services enabled by mesh networking have different requirements for the network. A lightbulb – switch pair would want low delay between flipping the switch and the light turning on. A temperature sensor might be less concerned about the delay but more interested in the ability of being able to send messages from one end of a building to the other. An alarm system would be highly concerned by the security of the network. Therefore, BLE-mesh should fulfil the requirement of different applications, while being simple to use by the end user. It does go a long way in effort to provide a wide range of features to the application developers. Some of the compelling reasons to use BLE-mesh for a product include: mesh technology, low cost chips (chips can be produced by independent suppliers), low power consumption, network security is default, smart phones are natively equipped with the hardware and Bluetooth is a trusted brand. However, BLE-mesh still have room for improvement. Specifically, configuration of the network can become a tedious task. The next section explains a significant issue which has not yet been addressed by the Bluetooth Special Interest Group.

2.5 Mesh network relay selection

A Bluetooth mesh network is managed by a Provisioner, which is a device providing the network administrator an interface to the network e.g. a smart phone or tablet. The Provisioner invites new devices into the network and configures them, allowing the devices to become nodes, that can utilize the mesh network. Provisioning is only happening point to point, meaning the Provisioner needs to be in range of the device being provisioned. The message system in the mesh network works with a publish and subscription method.

A node can publish a message to a unicast address and nodes subscribing to the address receive the message. When a new node is added to the network, it is assigned a unicast address for others to publish messages to. The Provisioner, upon configuration of a node, informs it about what addresses to subscribe to, this can be a unicast address or group address created by the Provisioner. The group address makes it possible for a single node to send a message to several devices at once. An illustration of how the addressing works can be seen in Figure 2-3Figure . When a network is configured, the Provisioner can leave the network.

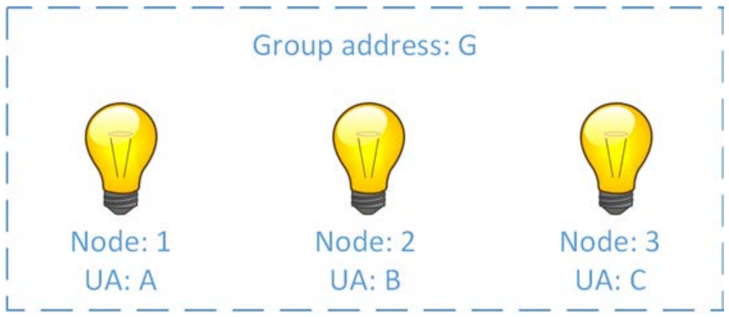

Figure 2-3 An overview of addressing and use of unicast addresses (UA)

A node can have several roles in the network. In the Bluetooth mesh network, there are four roles besides the Provisioner. Normal role is where the node has no special task in the network. A node can be low power, where it wants to consume as little energy as possible. The low power node achieves a low power consumption by sleeping most of the time. Low power nodes are often battery powered and therefore, by implementation default, obtain this role. When the node is sleeping, it cannot receive messages. Due to this it must team up with a friend node. A friend node can buffer messages for a sleeping low power node. If a low power node does not get a friend it will not be able to function in the network. The friend role is a role offered by nodes, which by implementation know this feature. The nodes able to be friends draw power from sources with endless or near endless power reserve and therefore, they have no problem listening to the network and forwarding the messages when the low power node is awake. The friendship establishment process between a low power node and a friend node is handled without the involvement of the Provisioner.

The last role is the relay role. Relaying nodes forward all received messages to extend the range of the network. The relay role is assigned by the Provisioner and this process can be very time consuming due to the need of setting up the relays manually to ensure the all nodes in the network are able to communicate with each other. Furthermore, it is a challenge for the human provisioning to choose which and how many of the nodes should be relays. If the Provisioner chooses too few, some of the nodes might not get the messages intended for them. If the Provisioner choose too many, the chances of packet collisions increase and thereby the efficiency of the network decreases.

This problem could be solved by implanting a self-configuration algorithm, which decides which devices should be relays. A simple Bluetooth mesh network can be seen in Figure 2-4Figure , consisting of the 4 different node types: Normal node, lower power node, friend node and relay node.

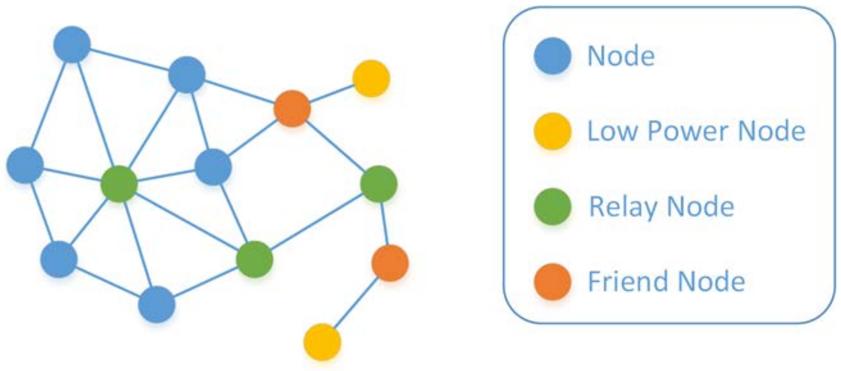

Figure 2-4 An example of a Bluetooth mesh network [8]

A mesh network without self-reconfiguration has a weakness. If a relay is moved or breaks, and the Provisioner is not aware, it can cause part of the network to disconnect. Therefore, it would be an improvement for the Bluetooth mesh network, if the network by itself could do configuration and reconfiguration of the relay role.

Clearly, the Provisioner has important tasks to do, both in the initial setup and configuration of the network, but also when changes happen to the current configuration. This means that for the network to function optimally, the Provisioner should be on standby 24h a day to step in and reconfigure the network. In the current model, where the provisioning tasks are carried out manually by a person, it is very infeasible to have a Provisioner on constant standby. This is a huge disadvantage in an otherwise autonomous network. Therefore, enabling self-configuration ability would be beneficial. The next section investigates current research in the field of relay selection.

2.6 Research investigation

A lot of researchers have addressed the development of algorithms, both centralized and decentralized, aimed at optimizing the way data is transferred in a wireless mesh networks. Essential for all algorithms is that they all makes a connected dominating set (CDS), as seen in Figure 2-5. A dominating set is a subset of nodes in the network, such that every node not part of the set has at least one neighbor from the set. Expanding on this, a connected dominating set is such that every node in the dominating set can reach every other node in the set via a route staying entirely within the dominating set. This is also referred to as a backbone network. Examples of dominating and connected dominating sets can be seen in Figure 2-5.

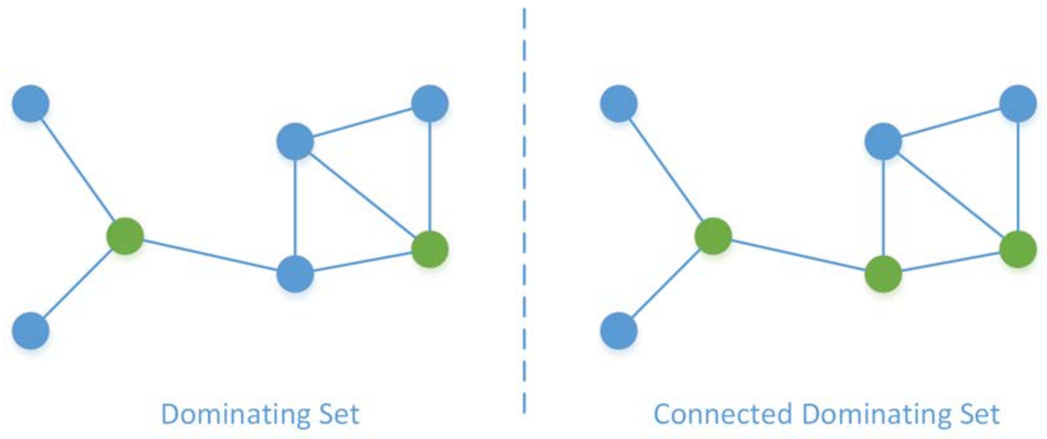

Figure 2-5 Examples of dominating sets. Green nodes are within the dominating set

A connected dominating set allows the nodes, outside the set, to stay in low power consuming mode for most of the time thus improving the power efficiency of the network. For the Bluetooth mesh relay selection problem, the connected dominating set would directly translate to the nodes which should enable the relay role. By reducing the numbers of nodes in the CDS, the probability of collision decreases at the cost of less redundancy in the network, against node malfunctioning.

To design our own algorithm, we investigate algorithms with different design focus. We have chosen to look into these focuses: fewest relays possible, Redundancy in the network and power efficiency.

2.6.1 Fewest relays possible

Creating a backbone network of nodes and keeping it as small as possible can reduce the chance of collisions occurring due to message retransmissions. This can be done by defining a dominating set that can cover the whole network with the least number of nodes. Such a set is called a minimum connected dominating set (MCDS) and finding it is a NP-hard problem. There are many ways of creating a MCDS and two of them is investigated.

The first approach is to create a CDS and then use reduction rules to reduce its size to a MCDS. The algorithm proposed in [9] is a two-phase algorithm that creates a dominating set of nodes and then reduces its size with what they refer to as "dominant pruning rule". In the first phase, each node checks if any of its neighbors are not connected to each other and if it has such neighbors, the node joins the dominating set. If the initial graph of the network is connected, the resulting set will always be connected too. The issue is that due to the simplicity of the selection the resulting set will almost always contain more than the optimal number of nodes needed to create a backbone that covers the whole network. Therefore, after the initial dominating set is defined the authors of the algorithm do a reduction of this set.

The algorithm removes a node from the dominating set if its neighbors are covered by a predefined number of dominating nodes. This is referred to as Rule k, where the parameter, k, specifies the number of linked nodes required to cover the neighbors of the selected node. The parameter k can be any number, higher or equal to 1, and the size of the dominating set decreases as the value of k increases. Increasing k though, also increases the complexity of the messages and the time for execution of the algorithm. This algorithm is most efficient in high density networks having unidirectional links between all the nodes.

The second approach is to create a MCDS by first defining an unconnected dominating set and then applying a rule to connect its members to form the MCDS. The algorithm proposed in [10] is compared to the first algorithm and aims to surpass its performance. It is also a two-phased algorithm where the first phase creates a dominating set which is then connected in the second phase. The algorithm uses color assignment in a greedy approach for creating a dominating set. The colors used are white, grey and black, where white stands for undominated node, grey means the node is dominated, but has white neighbors and black is for a node member of the dominating set that has no white neighbors. Initially all nodes in the network are colored white as none of them is dominated. In the first phase, each node exchanges the number of its white colored neighbors with its one-hop neighbors. The node with the highest count adds itself to the dominating set and is colored black while its neighbors are colored grey. This repeats until every node is dominated or is a member of the dominating set. The resulting set likely consisting of disconnected elements and phase two aims to connect those elements to form a connected dominating set. In phase two each node is assigned an id number. The algorithm then checks if any gray node connects nodes with different ids and if so it is colored black and joins the dominating set, thus creating a connected dominating set. The authors claim that the algorithm clearly outperforms the Rule k algorithm at lower ranges, but at higher ranges both algorithms experience the same performance, since the network is so dense that a few nodes are enough to cover the whole network. The algorithm though is not fully distributed. It is also unclear how the algorithm affects the lifetime of the network and the message complexity.

2.6.2 Redundancy in the network

By restricting the routing task to the CDS the network is made vulnerable to error, such as node malfunctioning, mobility or a node turning on and off frequently. This is because the CDS only preserves 1 connection to nodes not in the CDS. Thus, another design focus is redundancy. Depending on what kind of network you have it is important to maintain a certain degree of redundancy to the nodes not in the CDS. This is known as an k-connect m-dominating set(kmCDS).

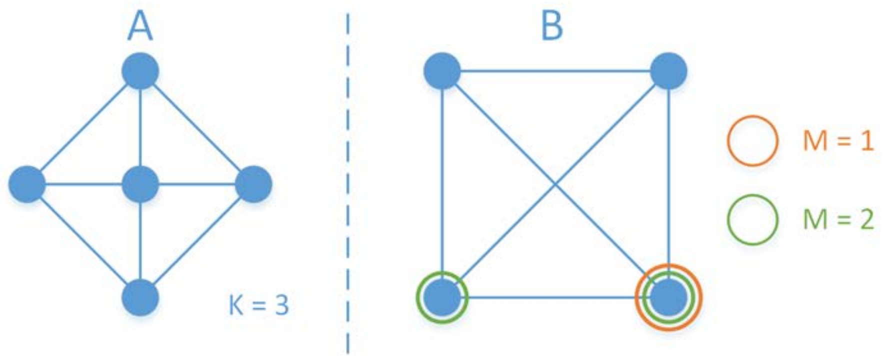

Figure 2-6 Example of k-connect m-dominating set.

Definition 1 [11]**:**
A network is k-connected if for each pair of nodes there exist at least k mutually independent paths connecting them. In other words, the network is still connected even after removal of k-1 nodes from the network, as seen in Figure 2-6-A.

Definition 2 [11]**:**
In a graph, $G(V,E)$, a node is said to dominate itself and all its neighbors. An m-dominating set D is a set $D \subseteq V$ such that every node in $V \backslash D$ is dominated by at least m nodes in D, as seen in Figure 2-6-B.

By combining these two definition the k-connect m-dominating set is defined:

Definition 3 [11]**:**
A set $C \subset V$ is k-connected m-dominating set (kmCDS) of graph, $G(V,E)$if the induced subgraph $G'(C,E')$ is k-connected and the set C is a m-dominating set of G

The algorithm proposed in [11], investigates how this can be implement with the assumption that it is possible in the graph. The algorithm obtains a m-dominating set by constructing a CDS and then makes the set kmCDS by adding nodes by following a complex ruleset. The article shows that it is possible to implement this in the real world, but the resource consumption like, network load, CPU load, time to finish etc. is unknown. Additionally, this algorithm is only applicable to networks, where a kmCDS is possible. How this is determined is also unknown, making it impractical for real life networks.

2.6.3 Power efficiency

When considering networks consisting primarily of battery powered devices, it is almost impossible not to consider ways to optimize the power consumption of the devices in the network. If a device with limited power supply is chosen to relay messages, the charge of its battery is depleted significantly faster. The algorithm proposed in [12] focuses on power efficiency by maximizing the lifetime of the dominating sets. The authors propose the idea of finding a maximum number of disjoint dominating sets. In graph theory, this is known as the "maximum domatic partition problem". The disjoint sets are activated consecutively allowing for most of the nodes to remain in low-energy mode thus prolonging the lifetime of the network. This is done to preserve some of the remaining charge in the devices part of the set. The algorithm picks nodes for the dominating set based on colors assigned with respect to the available battery charge for each device. The nodes are released from the dominating set when the residual charges reach a certain threshold to still allow for messages to be sent and reach the device. The challenge that this algorithm does not address is when there is no knowledge of the whole network, since all the assumptions made consider having full knowledge of the neighbors of each node in the network. Additionally, the paper does not provide any simulation results, since it is only described conceptually.

The presented research shared the commonality, that the results apply to ad hoc mesh networks in general. No research has considered implementation for specific technologies. Therefore, the knowledge gained from the investigated literature will be considered, as this project tries to solve the relay selection problem for BLE-mesh technology.

2.7 Problem statement

IoT devices are becoming more and more popular and therefore, there is need of having the right network technology to facilitate the IoT demands. A technology which could fulfill these demands is Bluetooth mesh. Bluetooth mesh uses Bluetooth low energy technology to facilitate a flooded mesh network. The Bluetooth mesh network is managed by a Provisioner that assigns the relay role in the network. The Provisioner must select the right number of relays otherwise the network will become unable to cover all nodes or message collisions might prevent sufficient communication. The Bluetooth mesh standard does not contain an algorithm, which decides who will be a relay, and therefore this is a task for the human doing the provisioning. When the provisioning is completed, the nodes in the network will have the same roles until the Provisioner reassigns them. This will create a problem if a node is moved or a connection breaks (due to a change in the propagation channel or failure of a node). This might result in a disconnection of part of the network, which can only be fixed when the Provisioner is reconnected. Self-configuration mesh networks have been of research interest for some time now. Particularly interesting for the relay selection problem is the problem of finding a CDS in any network. Effective algorithms focusing on different aspects of the CDS solution has been conceptualized. Implementing the algorithms in real life networks often present a challenge because some of the algorithms require full knowledge of the network. While the investigated literature provides valuable insight on how to tackle different problems of the creation of a CDS, none are perfectly suitable for the relay selection problem in Bluetooth mesh networks. Therefore, the rest of this report will be dedicated to solving the relay selection problem optimally for Bluetooth mesh networks. The goal of the project is to answer the following problem statement.

How could an algorithm for autonomous relay selection in Bluetooth mesh networks be designed so that it provides full network coverage and optimal message transmission conditions?

3 | Evaluation & verification

A crucial part of algorithm design and development consists of being able to verify and evaluate the algorithm. This chapter discusses and describes the methods and tools, which can be used for verification and evaluation. The network is represented as a graph $G(V,E)$ where the nodes are the V and the connections between them are called edges denoted with E.

3.1 Verification

A relay selection algorithms most important purpose is to select relays in a way that every node in a network can communicate with every other node. Any algorithm which fulfills this requirement is a valid solution. Any developed algorithm, should provide a connected dominating set (CDS). Although this might seem obvious and trivial, when it comes to networks and distributed algorithms the verification procedure might not be straight forward.

Different methods of verification exist. The most thorough are the formal verification techniques [13]. Formal verification utilizes mathematics to prove that some product (e.g. a piece of software or hardware) comply with the design specification. Formal verification requires a lot of modeling of the product to give mathematic guaranties. Therefore, formal verification is the preferred method for critical components. Relay selection in a mesh network in an office building or similar, is not critical enough to justify the formal verification. Therefore, there is a need for a less strict method of verification.

A more common verification technique involves thorough testing of the product, providing different test vectors to observe the behavior in different cases [14]. Verification of an algorithms ability to provide a CDS is straight forward, once the algorithm has been implemented. It then becomes a case of reproducing the test multiple time, varying the test vector:

1. Provide a test vector, which has a solution

2. Run the algorithm on the input

3. Verify the output

Verifying the algorithm by testing on different test vectors requires the test vectors to be representative for the entire problem space, which in the case of relay selection in arbitrary mesh networks is inexhaustible. Therefore, it can be useful to identify the parameters, which specify/describe the network, and provide edge cases for the tests. For the relay selection algorithm, the test vector consists of a connectivity graph of the network. The edges in the graph are weighted and are a representation of the quality of the connection.

The most descriptive parameter of a network is the average neighbor count of each node. This parameter influences the complexity of the relay selection problem, as more neighbors lead to more relay candidates. The neighbor count can be directly defined by two parameters:

- **Density**: The density of the nodes in the network pr. square meter is important, since denser networks means more neighbors. The density is varied by having constant area and change the number of nodes placed in the area.
- **Node range**: Node range is the maximum distance at which two nodes can communicate. The node range influence how many neighbors a node will have.

Tests need to be performed on these parameters, including combinations of the two. In addition to verifying the algorithm on networks with edge cases, i.e. very dense and very sparse networks, the algorithm should be verified on networks with values of the parameters which falls between the edge cases. This can be seen in Section 7.2.

3.2 Evaluation

Verification is the act of ensuring that the algorithm solves the solution. Evaluation is the act of determining how well the problem has been solved. To evaluate how good the solution is, it is necessary to specify the parameters on which to base the evaluation. The evaluation of the algorithm is split into two. One is the solution produced by the algorithm, and the other the resource consumption of the algorithm.

Firstly, the solution provided by the algorithm should be evaluated. Even though each parameter evaluates different aspect of the solution, they are still correlated with each other. The performance indicators of the solution are:

- **Collision**: Collisions are the main concern in the development of this project, based on the following argument. The solution is running in a wireless mesh network which provides neither collision avoidance nor collision detection. Combine this with flooding as the routing mechanism and power constraint devices, which want to save resources and therefore minimize the number of transmissions required. It becomes obvious that minimizing the number of collisions is an effective way of improving the probability that messages get through to the receiver with the minimal number of transmissions by the sender. Therefore, minimizing collisions is the focus in the design of the solution
- **Delivery probability**: Messages are transmitted with a purpose. Therefore, it is important that the messages are delivered to the specified recipient. The delivery probability specifies how likely it is that a message gets delivered to the desired destination. This indicator is directly correlated to the collision indicator stated above. When a collision occurs, the message is lost.

Some applications present on an office mesh network, might have a quite large tolerance to lost messages, while others like the ones in hospitals have a very low tolerance. If measurements from a temperature sensor are lost, it might not cause any trouble, but if measurements from a heart monitor are lost it could be detrimental.

- **Hop distance**: It is interesting to see if the solution provided generates a network where communications between some nodes are subject to an impractical route. Every hop in a network adds a delay. Additional hops translate into an increased risk of losing the message. The hop distance between two nodes forced by the solution, relative to the minimal hop distance is an indicator of how well the relays have been selected. If the hop distance increases so does the collision probability, since a packet is re-transmitter more to reach its destination.
- **Latency**: The latency of the message is the time between the source sending a message until the destination has received it. Two main factors impose latency to a mesh message: hop distance from source to destination and buffer time in intermediate relays. The mesh specification strongly suggests the upper bound on the transmitted number of messages should be 100 messages pr. 10 seconds. Therefore, if relays are loaded to a point where it has more message than it can transmit, it will naturally need to delay the messages further. It can also be the case where the buffer is overflowed and messages must be dropped.

Secondly, evaluating the algorithm itself is key, especially when the devices running the algorithm are resource constrained. Below is the list of performance indicators for a relay selection algorithm in mesh networks.

- **Configuration time**: The time used to produce a configuration from scratch is important, since it clearly sets some restrictions on the operability of the network. The measurement is done on a setup, where all nodes in the network are turned on simultaneously, meaning every node has zero prior knowledge of the network. The configuration time is measured as the time spent from turning on the nodes until the algorithm has decided for every node in the network. Individual algorithm might have better performance when devices are added gradually, or information about some of the network is known in advance.
- **Network load**: A relay selection algorithm must necessarily generate some network traffic to perform its duties. It is desirable that the network load imposed by the algorithm is done in a way that has least possible impact on the other operations of the network. Network load is also a function of the time spent on configuration. Depending on the requirements different applications have, their tolerance for the network load will vary.

- **Computational complexity**: Since the algorithm is running on BLE devices the computational complexity of an algorithm might increase the time and the power consumed by the hosting device.
- **Memory consumption**: The algorithm must be implemented on BLE devices which has limited memory. If the algorithm is using all the memory it can affect all other functionality on the node. This may prevent applications from running properly, thus affecting the relaying feature, which in turn affects the data flow through the network.

In design of distributed algorithms scalability is an important feature, because the algorithm should be able to be deployed in any size of network. The scalability is determined by two factors, first the scalability of the solution as function of neighbor count and secondly the scalability of the resources consumption by the algorithm pr. node. The scalability of the algorithm can be inferred by the evaluation of the above-mentioned parameters.

The evaluation of the algorithm should happen in a random scenario, because the algorithm must be able to handle different scenarios instead of just one static. The random scenario is a free space area with constant size, where the node density and node range can be varied. The reason for free space is that obstacles will only change how the nodes connects.

Like any engineering problem, the issue of selecting relays does not have a perfect solution. Every solution includes some tradeoffs, and how each parameter is weight may change in different scenarios. For this report, the focus in terms of design is collision minimization. The designed algorithm will be verified and evaluated as stated in the chapter. But as the collisions is the design focus this will be evaluated deeper than the other parameters. Verification and evaluation results is presented and discussed in chapter 7 |. Verification and evaluation will be carried out by using a simulator.

3.3 Network simulation

Network simulation has been established as the method of choice for verification and evaluation on algorithms destined for network operations. This is the case because computer networks quickly become too complex for traditional analytical methods. Simulation, however, still relies heavily on mathematical modelling [15]. To simplify the model, it is assumed that every nodes' range is circular. Due to this the network is represented as a unit disc graph (UDG) as illustrated in Figure 3-1.

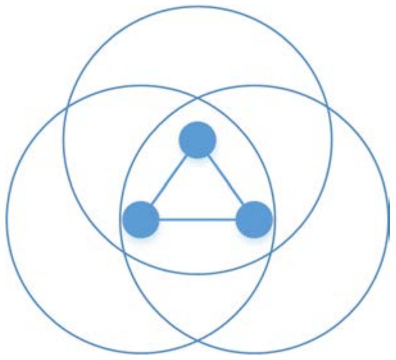

Figure 3-1 A example of a unit disk graph (UDG)

Each element of the computer network is simulated using an appropriate model. Path loss is simulated based on some empirically obtained mathematical model. Noise and interference is added to each simulated transmission, according to a specified model, and so on. By doing so, the result is a simulator which can produce simulated conditions which are highly realistic and easily scalable. Another key feature of computer simulation of networks is the reproducibility. Evaluating different algorithms subject to the exact same network conditions becomes possible.

The biggest limitation of a network simulation is the fact that the simulation is only as good as the models it is based on. Therefore, it is important to analyze and understand the conditions under which the product is being deployed. Analyzing transmission power control algorithms for indoor mesh networks, using a free space propagation model might not be suitable and produce results which are not reproducible in the real world. On the other hand, some scenarios might be so complicated, that modeling would present a huge work load. In such cases, implementation on a real-life test bed might be a better option. However, verification and evaluation for the purposes of this report is carried out by network simulations.

3.4 Simulation tools

The most common type of network simulator is discrete event simulator. A discrete event simulator processes only special events in discrete time. When an event has been executed time is fast forwarded directly to the point where the following event is starting. The main mechanism for organizing events is the event queue. This is suitable for network simulators as this give the option only to simulate what happens when data are sent, and skip the time slots where the network is not moving data. Discrete event simulation can be created in different environments or as standalone programs.

This project uses two simulation tools, MATLAB (**mat**rix **lab**oratory) and ns-3. MATLAB is chosen for its high abstraction level and ns-3 for its realistic network simulations. The tools will now be described further.

3.4.1 MATLAB

MATLAB is a numerical computing programming language (as opposed to a symbolic environment), with a wide and powerful variety of available functions used in linear algebra, signal processing, image processing, etc. MATLAB is a high-level abstraction language, making it fast to develop software concepts. MATLAB's ease of use has made it the language of choice for two tasks in this project: Verification and initial solution evaluation.

Verification of relay selection is simple. The network is represented by a graph. If two nodes are in range of each other’s, they are connected. However, two nodes which are not in range of each other are only connected if they are both connected to a CDS of relay nodes. To verify the CDS, check if all nodes are dominated by at least one member of the dominating set and then if there is a path between all the members of the dominating set.

MATLAB is similarly used for initial evaluation of the relay selection algorithm. More specifically it is used for a sanity check of the solution which the algorithm produces. The check involves only looking at the selected nodes – not simulations of traffic or any other network related metrics. To minimize collisions, the solution is simply evaluated based on the number of nodes, it selects as relays (and density hereof), as well as their placement relative to each other. The goal is a low density of relays and a substantial spacing between relays - while maintaining a connected network.

MATLAB is the tool of choice for these two tasks, because MATLAB enables fast development and tweaking of the algorithm in early design stages.

3.4.2 ns-3

MATLAB provides simple evaluation of the algorithms solution, but to get a more extensive evaluation of the beforementioned performance indicators, a more complex network simulator is needed. Numerous possibilities exist. The simulator of choice for this project is ns-3 for three main reasons. First ns-3 is a complex simulator which is highly customizable in its models of the real world. Thereby ns-3 enables realistic simulations. Second ns-3 is an opensource project free for anyone to use. This has fostered a significant community of developers and a reasonable documentation on how to use the simulator. Third ns-3 is primarily targeted the research and educational community.

Ns-3 is used for the extensive evaluation of the algorithm. Simulation of the behavior of BLE devices makes is important to provide insight to how the algorithm would operate in real life implementations. Behavior like a BLE radios half duplex capabilities, and restrictions on how often and how fast the device can transmit, is captured and simulated in ns-3

Unfortunately, BLE and BLE-mesh are not default implementations in ns-3. Therefore, capturing the behavior of these protocols requires a development effort. Fortunately, BLE simulation has caught the interest of others, including Samsung. Samsung managed to implement the basic features for BLE in ns-3 required for BLE-mesh. Implementation of BLE-mesh was carried out as a collaborative effort between the authors of this report and the computer science student group at Aalborg University, group d602f17.

Parts of the BLE-mesh standard have been implemented to support the functionality for relay selection. This includes, among other things omission of everything related to security in the specification. Protection against bit errors and other packet corruption has also not been implemented, because the BLE implementation does not introduce these errors. For an overview of the features of the mesh stack see Section 2.4. The implemented simulator consists of several modules for which some of them are not developed in this project. An overview of who developed what layers can be seen in Table 3-1. The table includes a description of the implemented features in each layer and the authors insight into the specific way things have been implemented.

Layer	Developer	Implemented features
Model layer	17gr820 White box	Contains the implementation of our own algorithm.
Foundation model layer	Not implemented	
Access layer	17gr820 White box	Models can attach a callback, opcode and mesh address. Incoming messages will be forwarded to the correct model. Outgoing messages will be attached a header with opcode and handed to the upper transport layer.
Upper transport layer	17gr820 White box	An empty trailer (which should contain integrity check) is added and removed in this layer. Incoming messages will be forwarded to the access layer. Outgoing messages will be save in a queue. There is one queue per destination. Only one reliable message at a time will be transmitted per destination. Next packet for a destination is released when the lower transport layer signals that the message has been delivered or canceled.
Lower transport layer	d602f17 Grey box	Segmented and unsegmented access messages is implemented. Segment tracking is done based on a unique identifier. The layer is only able to receive segments from one source at a time. Segments are acknowledged. The upper layer is informed on the outcome of the transmission. A header is added and removed in this layer.

Network layer	17gr820 White box	Source, destination, TTL and sequence number is added in a header. The layer maintains a cache with the 10 most recent packets. If an incoming packet is in the cache it is not processed. If the relay functionality is enabled, packets will be relayed within 20ms (uniformly distributed). No header obscurity or other security measure is performed
Bearer layer	17gr820 White box	The header is zero stuffed to match the size in the specification. Outgoing messages are handed to the BLE core with a minimum of 20ms spacing. This layer is managing when the BLE core is advertising and scanning.
Bluetooth low energy core	Samsung Black box	The BLE core handles transmission of the unconnectable indirect advertising messages used by BLE-mesh. Transmission on all 3 advertising channels and scanning is performed here.

Table 3-1: Overview of the development on each layer

Simulations in ns-3 will be close to real life, but has some short comings. The path loss model is free space, meaning no building walls or other reflective surfaces are simulated. Despite this simplification, the results from the simulation will not be significantly different. The path loss model will have an impact on how signals travel. Therefore, from node perspective this only changes the RSSI, which results in different interpretation of the distance between the nodes, changing the topology of the network. Since simulations are mainly done in random networks, it is not a significant factor, as different path loss model might just result in another random network topology.

Similarly, to the lack of surrounding walls, the initial simulation is based on an environment where presence of interfering equipment is disregarded. For office environments, it is highly likely that other devices will interfere with the BLE-mesh network, e.g. Wi-Fi.

Because the BLE and BLE-mesh protocol stacks are only partly implemented, no effort has gone into implementing error detection. This means that introducing bit errors to transmitted packets will stay undetected all the way op the protocol stacks. Therefore, no bit errors are implemented in the simulator. Additionally, no packet errors are introduced, rendering transmission success a function of only distance between source and destination. This can, however easily be reconfigured, if evaluation of the algorithm under noisy conditions becomes desirable.

4 | Relay selection model

Now that the verification and evaluation procedures have been settled, the design of an algorithm begins. This chapter presents a framework for the algorithm. The framework is presented as a conceptual model, containing multiple phases. In BLE-mesh applications are placed in the model layer. The designed algorithm must be implemented as a BLE-mesh model. This impose several constraints on the algorithm, as all interactions happen through the access layer. The rest of the chapter contains an explanation of the conceptual model, and a limitation of what phases are being implemented in the timeframe of this project.

4.1 Conceptual model description

To design and implement a functional relay selection it has been chosen to design a conceptual model that divides the algorithm into difference phases. The phases are independent modules, which can be developed in parallel. Furthermore, the independence between phases makes it possible to change the implementation of a phase without affecting the other phases, if the interface is maintained. The model has been divided into five phases, each phase with its own specified task to carry out, before the next phase is started. The algorithm circles through the phases, and loops around after the 5th phase. The model spends most of its time in the phase network monitoring waiting to get activated. The model is designed in such a way that allows it to run at the same time as other BLE-mesh models. The conceptual model design keeps the old network topology and its functionality until the nodes have decided to be a relay or not. Thus, only the broken part of the network is affected until the model has reconfigured the network. The model and its phases can be seen in Figure 4-1. The phases are described below.

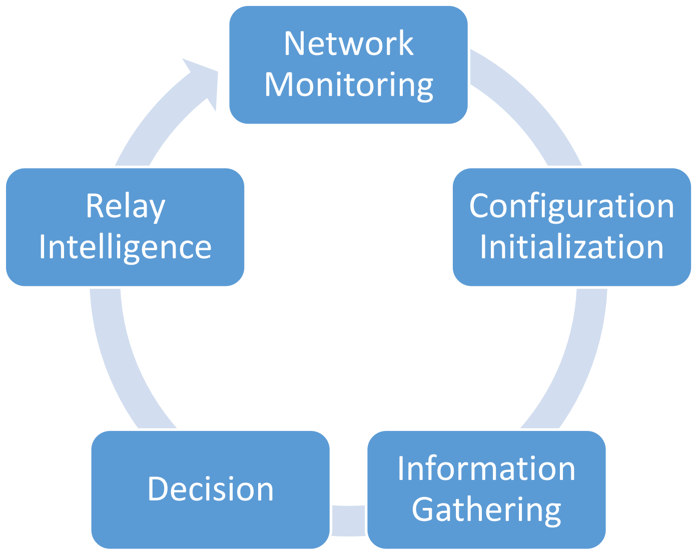

Figure 4-1 Overview of phases

Network monitoring phase: This is the phase in which a node can decide to trigger a self-configuration of the network. A node spends most of its life in this phase. The network monitoring phase contains all the intelligence about when to trigger. Several reasons can lead to a self-configuration of the network. The self-healing intelligence could have detected a break in the network or the Provisioner could have asked the node to start a configuration. When a node triggers, it proceeds to the configuration initialization phase, and it can no longer trigger a new self-configuration until it returns to this phase.

Configuration initialization phase: This phase informs all the nodes in the network, that a self-configuration of the relays is about to start. The node who triggered sends (floods) a message to all nodes in the network informing them that a self-configuration is starting. When a node receives this message, it must ensure that only one configuration is running. If several nodes decide to trigger at the same time, it could lead to conflicts. Furthermore, this phase ensures that a broken node can't initialize a self-configuration more than once for a given time span. For example, a broken node keeps rebooting and after each reboot it tries to configure the network. This conflict should be solved in such a way, that the broken node can only trigger once, and is otherwise ignored. When a node has solved eventual conflicts, it proceeds to the information gathering phase.

Information gathering phase: The information gathering phase gathers all the information needed for the decision phase. When a node enters this phase, it collects information about its surrounding neighbors. The phase has a parameter called network visibility, which defines how many hops away the node shall collect information from. A visual interpretation of the network visibility can be seen in Figure 4-2. When information about the network has been collected, the node waits for its surrounding nodes to finish and then proceeds to the decision phase.

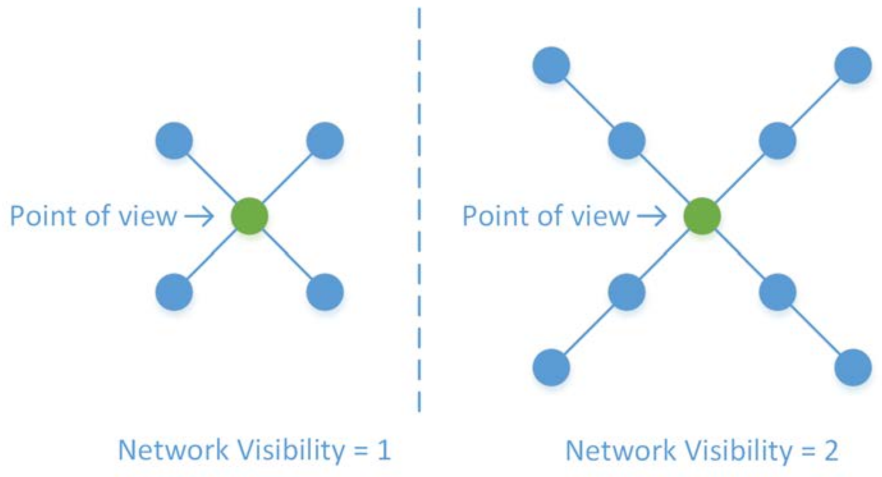

Figure 4-2 Network visibility example

Decision phase: In this phase, the node uses the information collected in the information gathering phase to decide if it should be a relay or not. The only demand for this phase is that it selects the relays such that they form a CDS. When the network has decided, which nodes should be relays, the nodes chosen not to be relays reenable their trigger and returns to the network monitoring phase. If the node has been chosen to be a relay, it proceeds to the relay intelligence phase.

Relay intelligence: The relay intelligence phase is an extension of the trigger phase. A node in this phase maintains all functionality from the network monitoring phase. In this phase, the relay nodes have the option to communicate with each other to increase their knowledge of the network. This knowledge can be used to do intelligent decisions such as routing.

Due to the time limitations for this project, some features of each phase have not been implemented. The next section presents a description of the features that are not implemented and how the missing features are worked around.

4.2 Limitation of the conceptual model

Due to the time constraints of the project and its focus, the following phases and functionalities are not implemented: the network monitoring phase, the conflict handling functionality in the configuration phase and the relay intelligence phase. Since the configuration initialization still needs to be initialized, a simple work around has been implemented. The phase is triggered as if the Provisioner has ordered a node to start the configuration.

Manually triggering the configuration initialization phase introduces another limitation. Having the phase triggered by a specific node, there is no need for the conflict handling functionality, since no other node tries to start the configuration initialization. Furthermore, the relay intelligence phase is not implemented, and all nodes returns to the network monitoring phase after finishing the decision phase.

5 | Design of the relay selection

This chapter describes how the configuration initialization phase, information gathering phase and decision phase have been designed.

5.1 Configuration initialization

Due to the limitations described in Section 4.2, the only task for a node in this phase is to inform all its neighbors that a relay selection is about to start. The phase is either started by a manual trigger from the Provisioner, or by the node receiving a configuration initialization message. Assuming the node has no prior information about the network, the only way it can communicate with its neighbors is to broadcast out a message saying, " we are about to start the relay selection". The message is rebroadcast by every node, to ensure that the entire network is aware that configuration has started. To avoid the possibility of multiple nodes broadcasting at the same time, a random delay between the broadcasts from each node is introduced. Managing this delay is important since it directly affects the runtime of the phase. The larger the delay the lower the chance of collisions, but that also means the configuration initialization phase will take longer. Since the triggering node has no knowledge of the rest of the network it cannot know when all nodes have proceeded into the next phase. Instead, the node waits for a certain time before moving into the information gathering phase. The length of this waiting time also increases the duration of the phase so it should be managed carefully.

5.2 Information gathering

The information gathering phase must collect the information needed for the decision phase. When a node enters this phase, it has no prior information about the network and everything must be collected from scratch. The more knowledge a node can have about physical constraints of the environment the better decision it can make. The only parameter about the physical environment supplied by BLE is the received signal strength indicator (RSSI). This number indicates how strong the signal is at a link between two nodes. RSSI can in an ideal environment be translated into a function of distance. An office scenario is far from ideal as there is reflection and shadowing of the signal, which can cause large fluctuations to the RSSI. Even though the assumption that RSSI translates into distance might not hold, it is the best guess a node can make about its placement compared to other nodes. Therefore, a node wants to collect information about the RSSI. For a node to be a good relay it needs to keep its position for a longer period, because if a relay node moves it might disconnect part of the network. Therefore, a node also needs to obtain information about how likely another node is to move. This information does not exist within Bluetooth. This leads to the introduction of a new parameter which in this report is called the movement probability.

Movement probability is a static number encoded in a node by the manufacturer. The number describes how likely the node is to move. A bulb will have a low probability as it is considered static and a mobile phone has a high number as it very likely that it will move. The number is in the range one to ten. One is a very static node and ten is a node which is very likely to move. It is up to the manufacturer to set this number. Therefore, the information gathering phase need to collect these two parameters from all nodes with the network visibility range. For more information about the network visibility see Section 4.1.

The information gathering can be divided into two sub parts, which in the report is referred to as: discovery and information from neighbors. Discovery handles how the node identify other nodes within its range and information from neighbors is about getting information from the discovered nodes.

5.2.1 Discovery

If a configuration is running it could be caused by two things: the network has changed or the network is configuring for the first time. Therefore, a node cannot trust the knowledge it has, and it needs to (re)discover the network. A node should broadcast a message informing others of its presence several times, to increase the delivery probability of the packet. This message is called "here I am". The node must listen for broadcast packets from its neighbors. The packet should contain the movement probability and the RSSI can be measured from the packet when received. If the received packets RSSI is beneath some threshold, it is not registered as a valid connection and the source node is not seen as a neighbor. All nodes shall maintain their own network knowledge list, where information about the network is stored. From this information construction of a connectivity matrix is possible. When a node receives a configuration message, it saves the unicast address of the node who published the messages and the payload, consisting of the movement probability, and the RSSI of the received messages in the knowledge list. To save memory the movement probability and RSSI are mapped into a single byte in the knowledge list. This number is called the relay likelihood.

$$RelayLikelihood = 255 - \left(\frac{100 + RSSI}{10} \cdot MovementPropability\right)$$

The relay likelihood is a number taking up 1 byte of memory, and if the number is large, it is likely that the node becomes a relay.

When a message has been received, the node updates its knowledge list, which can be seen in Table 5-1. The node sets the field "connected through" to the unassigned addressed used in BLE-mesh, and the payload is saved in the field relay likelihood. The field Unicast address of node, is set to the source address from the message. Each field in the knowledge list represents a connection between two nodes and its size is five bytes, since a mesh address is 2 bytes. A connection is only saved once, so if A to B is already in the list B to A would not be saved.

Field name	Unicast address of node (2 bytes)	relay likelihood (1 byte)	Connected through (2 bytes)
Field description	The unicast address of the element added to the list	From 0 to 255	This value is set to unassigned address if the node is connected directly to the node maintaining the list.

Table 5-1: Network knowledge list

When a node has broadcasted its "here I am" message, it must ensure nodes in its visibility range have finished collecting data about its neighbors before preceding to the next sub phase. Therefore, the node should wait a random amount of time and then go to the next sub phase, information from neighbor.

5.2.2 Information from neighbor

Nodes in in this phase collect information about nodes in the network in the distance specified by the network visibility, see Section 4.1. If the network visibility is set to one this sub phase is not used. If it is set to more than one, the node needs to collect information. The nodes send a "request view" messages as reliable unicast messages to their neighbors, one at a time. When a node receives the "request view" message it replies with a message containing all nodes it can see, found in the first sub phases. In the replay, the unicast addresses of the neighbors and their relay likelihood are packed into one payload. One neighbor node takes up three bytes in the payload. The maximum allowed size of a message that can be passed to BLE-mesh is 377 bytes. This makes it possible to send information about a maximum of 125 neighbors in one message. When a node receives an answer to a "request view" messages, it is saved in the knowledge list, seen in Table 5-1. The node saves the unicast address of the node who sent the replay as the "connected through" value, and then saves the received payload as seen in Figure 5-1.

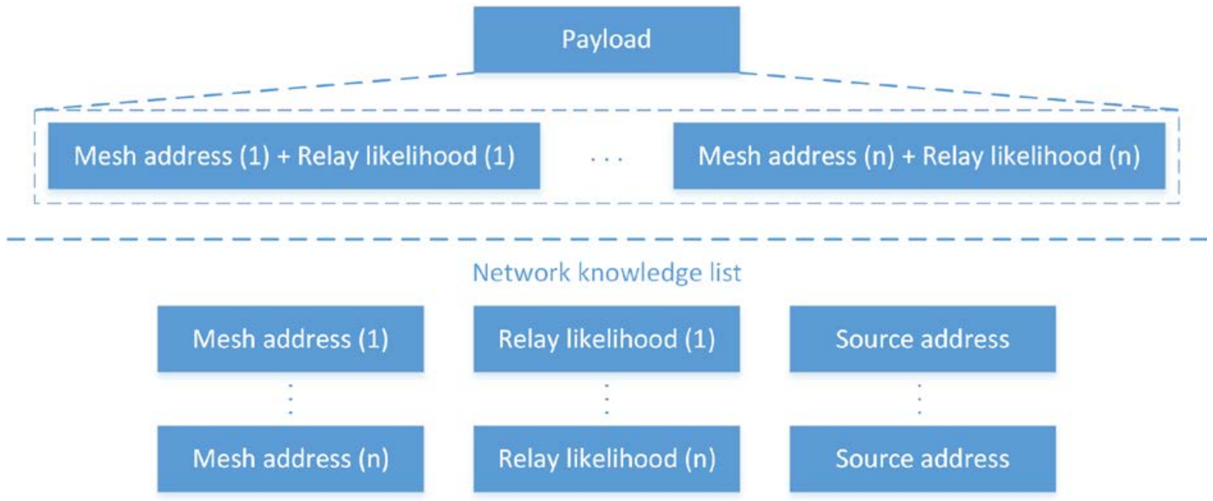

Figure 5-1 How a payload is saved in the knowledge list

If the network visibility asks for more hops the node will now request the node, it just received information from, about their neighbors and continue until the network visibility number is reached. When the information gathering is done, the node will proceed to the decision phase.

5.3 Decision

When a node has collected the needed information in the information gathering phase the decision phase starts. The goal of this phase is to decide if a node should be a relay or not, and then publish the decision. As described in the evaluation chapter the focus of the algorithm is to minimize the probability of collisions by selecting as few relays as possible and thereby forming a CDS. A CDS with few number of relays is not necessarily a MCDS. Finding a MCDS is an NP-hard problem (for an explanation of problem hardness refer to the Appendix D, where the NP-complete problem of finding the minimum vertex cover of a graph is covered). Therefore, it is not a design goal to solve the MCDS problem, but generate a CDS which contain few relays.

The approach to designing the appropriate algorithm is to iteratively design and evaluate an algorithm, making the algorithm better in each iteration. The starting point is assuming that the nodes have full knowledge of the network. Throughout the iterations the knowledge is reduced to the amount of information available from the information gathering phase. Two iterations have been performed assuming full knowledge of the network. These iterations are referred to as centralized algorithms. The last iteration assumes limited knowledge about the network, and is referred to as the distributed algorithm

5.3.1 Centralized algorithm

The centralized algorithm assumes full knowledge of all nodes in the network. The first step for the algorithm is to reduce the connectivity graph to a tree structure, such that it is possible to see which nodes connect other nodes as seen in Figure 5-2.

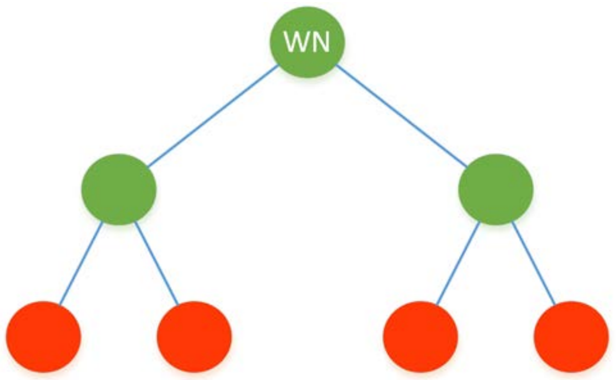

Figure 5-2 A tree created with shortest path from work node (WN) to every other node

In the tree, all leaf nodes should not be a relay since they only have one edge, meaning they do not add any new nodes. Nodes with more than one edge are selected as relays, as seen in Figure 5-2Figure , were red marked nodes are not relays and the green are relays.

The first iteration of the algorithm is illustrated in Figure 5-3 as a flowchart and in Appendix A as a visualization of how the relays have been selected.

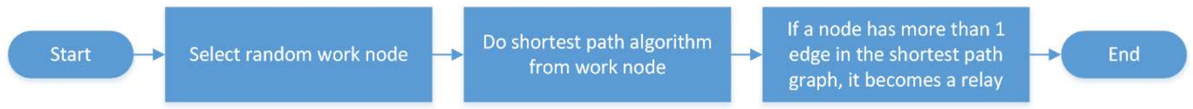
Start
Select random work node
Do shortest path algorithm from work node
If a node has more than 1 edge in the shortest path graph, it becomes a relay
End

Figure 5-3 Flowchart of the first iteration of the algorithm

In the algorithm, a random work node is selected, to emulate a random node in the network triggering the self-configuration. From this node, a shortest path algorithm is used to transform the graph into a tree. The shortest path tree has been chosen over the minimum spanning tree because the shortest path tree produces a more direct path to each node. An example of the shortest path versus minimum spanning tree is illustrated in Figure 5-4. From the example, it is possible to see how shortest path tree gives the least number of relays selected. The tree structure ensures a CDS.

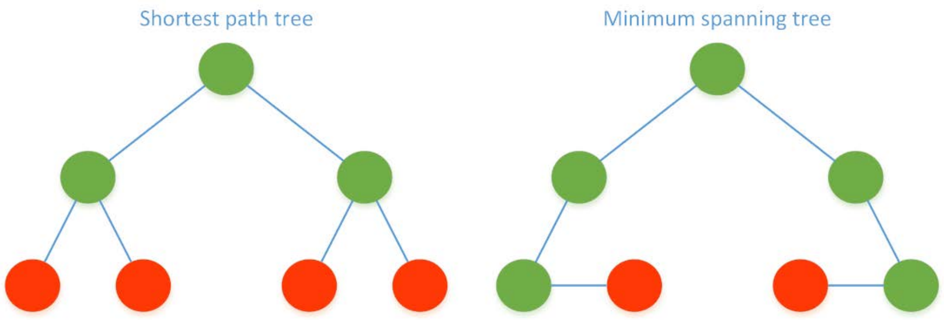

Figure 5-4 Shortest path tree versus minimum spanning tree

All simulations in this chapter are performed on a constant area of 50 x 50 meters and a constant node range of 10 meters is executed. The node density is increased in steps of 0.1. At each step, 10 random networks are simulated to compensate for the diversity of different networks, and from these a mean value is found. The simulated area is 2.500m^2 and each node can, with its range of 10m, cover approximately 314m^2. The simulation of the centralized algorithm can be seen in Figure 5-5Figure . The desired outcome is that the number of relays decrease as a function of node range and keeps a static level when the grid size has been covered.

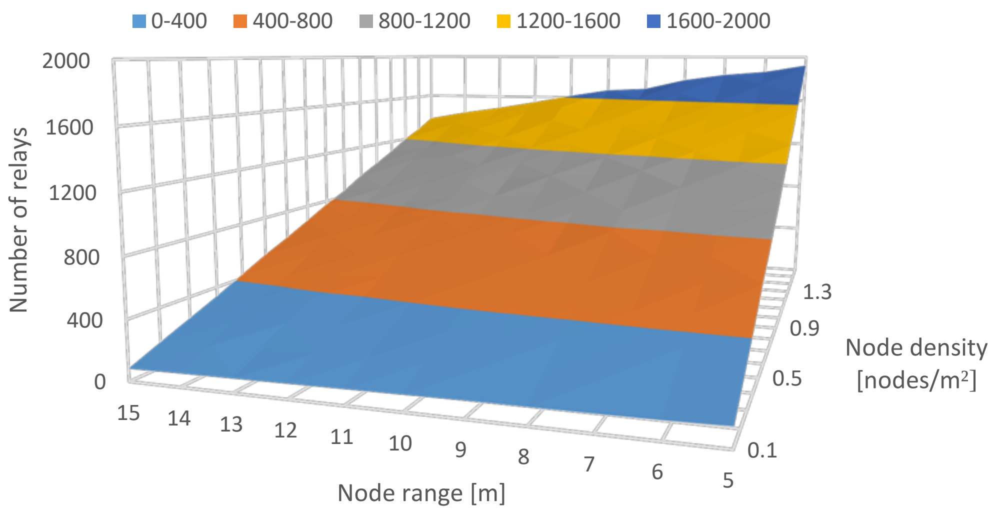

Figure 5-5 Simulation of shorts path without logic

The simulation seen in Figure 5-5, shows the number of nodes selected as relays decrease as a function of the node range. This is a desired property as it shows the algorithm is aware, when a node can cover a larger area, fewer relays are needed to form a CDS for the network. Furthermore, it is seen the number of relays increases linearly as a function of node density, which is not a desired feature, since when an area has been covered, more relays are not needed. Preferably, the number of relays should stabilize once the CDS of relays covers the entire simulation area. Therefore, extra logic is added to the algorithm in iteration two.

The idea of adding logic to the algorithm is to use the same approach but instead of choosing all nodes with more than one edge as relay, the logic adds some intelligence to the nodes with more than one edge. To simplify the problem, iteration two of the algorithm does not work with the whole graph at once, but instead works with one node at a time and then traverses the graph. This gives a node the ability to do a decision and then the other nodes can make their decision with the former decision in mind. The node still has full knowledge of the network and after it has made its decision it passes a token to the next node, which has full knowledge of the network state.

The second iteration of the algorithm starts by selecting a random node to emulate a random trigger in the network. The randomly selected node is called the work node. The work node is the first selected relay and it is saved in the relay list. From the work node, the shortest path is found to every node in a two-hop range. Based on this, the algorithm finds all neighbors with only one edge and marks them as nodes that cannot relay. Then all edges to the nodes marked as cannot relay are removed from the graph. The remaining neighbors are marked as maybe relays. An example of this can be seen in Figure 5-6.

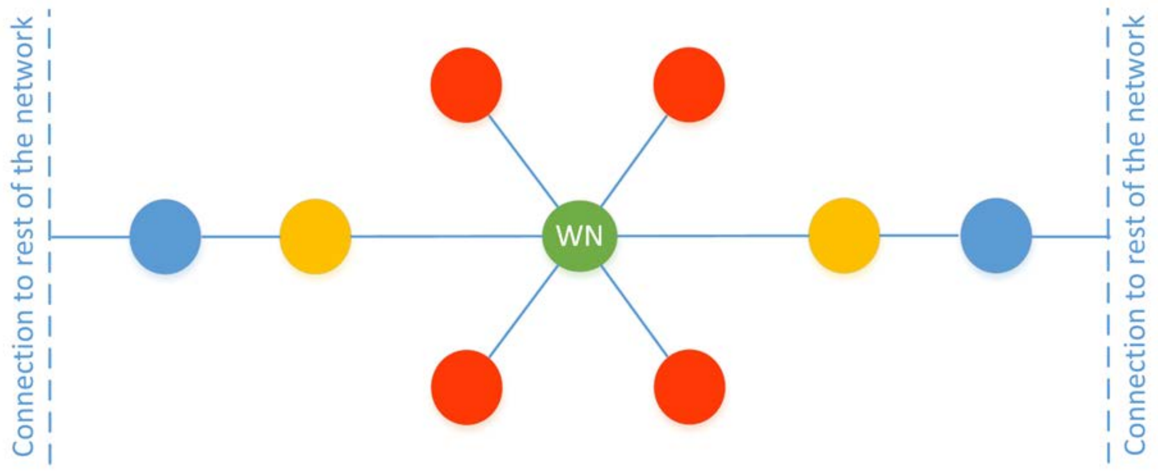
Connection to rest of the network
WN
Connection to rest of the network

Figure 5-6 Nodes within range of the work node after marking. Red is cannot relay, yellow is maybe relay and green is relay

The next step for the algorithm is to decide which one of the maybe relays to select. Here the algorithm branches as there are two options:

Option one, the algorithm chooses the maybe relay node with the largest edge weight (relay likelihood) as the new work node. Selecting the maybe relay with the largest edge value ensures the selected relay has a low probability for moving and is as far away as possible to give a larger network coverage. When the algorithm reaches a dead end in the network, it goes to option two.

Option two, the algorithm has reached a dead end and must go back in the maybe list to find a new work node to start from. Therefore, the algorithm does a cleanup of the maybe relays to ensure all marked nodes are a candidate to be a relay node. Nodes in the maybe relay list is not a candidate if they only connect to already selected relay's or don't connect any new nodes. In this case, the maybe relay is changed to a cannot relay and removed from the graph. When the list has been cleaned, the algorithm chooses the maybe relay node with the lowest weight when saved in the maybe relay list. The chosen node is the new work node, and the algorithm goes to option one. When the maybe list is empty, the whole graph has been traversed and the algorithm is done. The flow of the algorithm is illustrated as a flowchart in Figure 5-7.

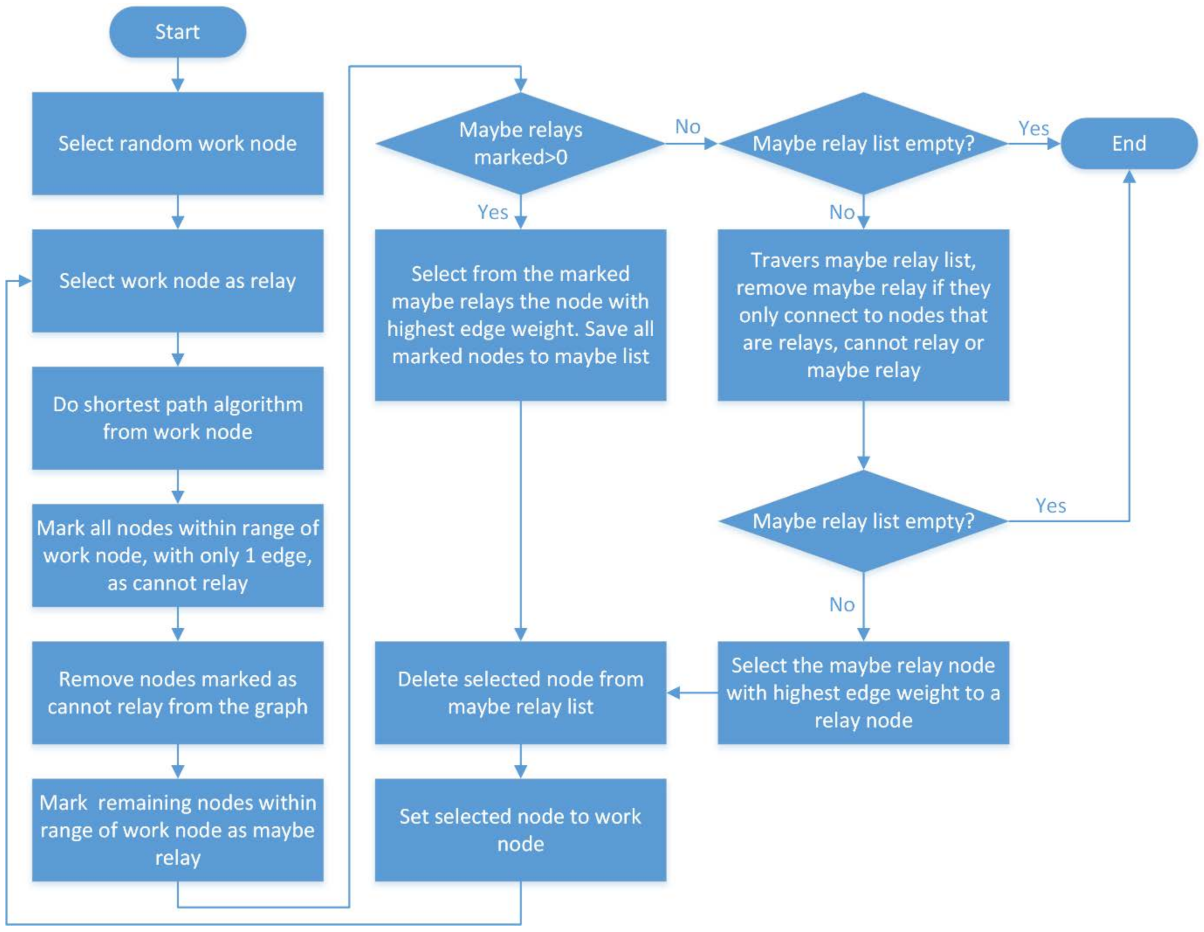
Start
Select random work node
Select work node as relay
Do shortest path algorithm from work node
Mark all nodes within range of work node, with only 1 edge, as cannot relay
Remove nodes marked as cannot relay from the graph
Mark remaining nodes within range of work node as maybe relay
Maybe relays marked>0
Yes
No
Select from the marked maybe relays the node with highest edge weight. Save all marked nodes to maybe list
Delete selected node from maybe relay list
Set selected node to work node
Maybe relay list empty?
Yes
End
No
Travers maybe relay list, remove maybe relay if they only connect to nodes that are relays, cannot relay or maybe relay
Maybe relay list empty?
Yes
No
Select the maybe relay node with highest edge weight to a relay node

Figure 5-7 Flow chart of algorithm iteration two

The algorithm has been simulated to evaluate its performance. The results from the simulation can be seen in Figure 5-8. The figure displays the number of relays decreases as a function of node range and the number of relays is stable when the density increases.

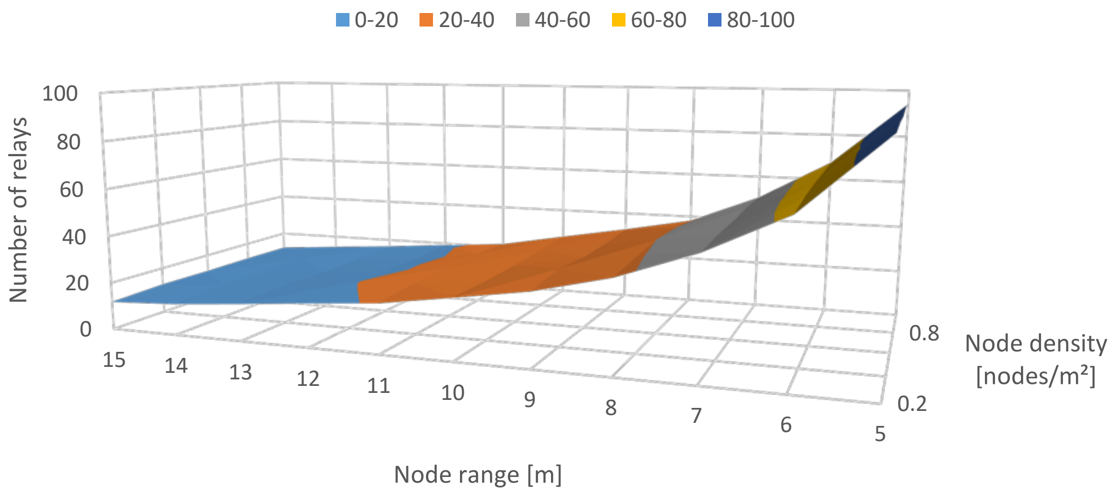

Figure 5-8 Number of relays selected by iteration two of the algorithm

An interesting property of the algorithm is when the node density increases the algorithm stabilizes and uses the same number of relays to cover the same area. Therefore, it can be concluded iteration two has the desired properties. Furthermore, the algorithm is verified to provide a CDS for all simulated scenarios.

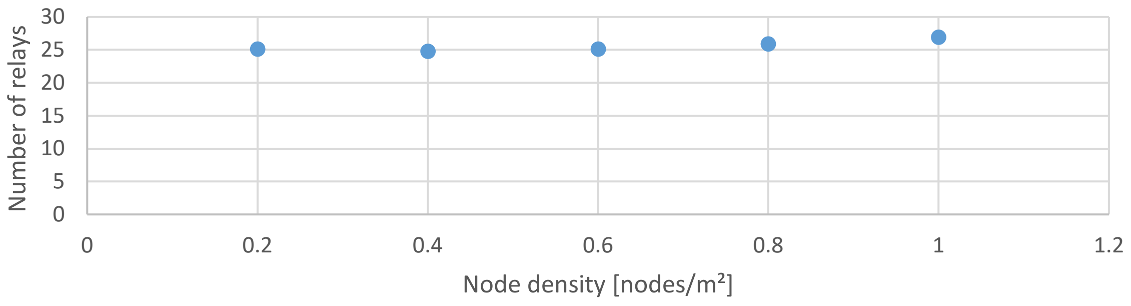

Figure 5-9 Number of relays selected by iteration two, when node density is varied

A realization of relay selection is illustrated in a figure in Appendix A. Inspection of the figure reveals an undesired side effect of minimizing the number of relays. Minimizing the relays will likely avoid cycles in the network. This is a problem, because it can result in suboptimal delivery routes of messages between some nodes. The problem is illustrated in Figure 5-10. When node four wants to communicate with node one it must go through three hops instead of two, due to the broken cycle. The network delay and the chance of getting a collision increases, because the message must travel through more hops. Introducing more cycles to the network provides redundancy to the messages and shortens the hop distance between sender and receiver. Therefore, the next iteration will be allowed to select more delays to avoid these suboptimal delivery routes.

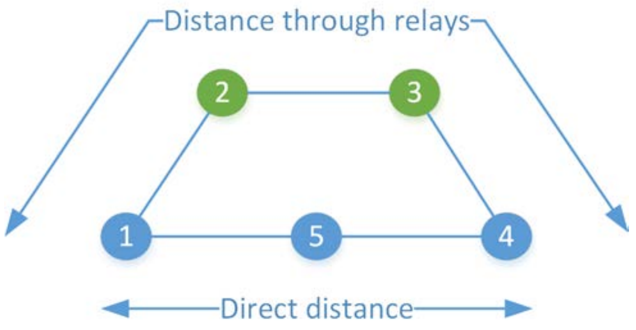

Figure 5-10 A example of a broken cycle. Green nodes are relays

A scalable centralized algorithm capable of doing self-configuration of the network has been designed. The algorithm has full knowledge of the network, which in this report is not deemed a realistic scenario for real life networks as this knowledge is very hard to collect for large networks with moving nodes. Therefore, the distributed algorithm is designed with the ability to provide a solution when the network visibility is low.

5.3.2 Distributed algorithm

The information available in the decision phase is the one collected from the information gathering phase, the span of which is specified by the network visibility. This limits the knowledge available to a node compared to the centralized algorithm. Therefore, the design of iteration three of the algorithm will have knowledge equal to a network visibility of two. This means a node can see all its neighbor and their neighbors.

The distributed algorithm starts at the node which triggered the configuration initialization phase. This node is called the work node. The working node has the decision token and it does a shortest path algorithm on the two hops of visibility it has. All nodes that are its neighbors and only have one edge do not qualify to be a relay, exactly as in the centralized algorithm. If a node has more than one edge it connects more nodes to the network. It is important that the one-edge-check only happens for the neighbors of the work node, as all nodes beyond its immediate neighbors appear to only have one edge, even though they connect more of the network. This is because of the visibility specified by the parameter network visibility.

The nodes identified with only one edge in the shortest path tree are not aware of the fact. If they are asked to do a decision they create different shortest path from their own perspective, where they are likely to take on the relay role, see Figure 5-11.

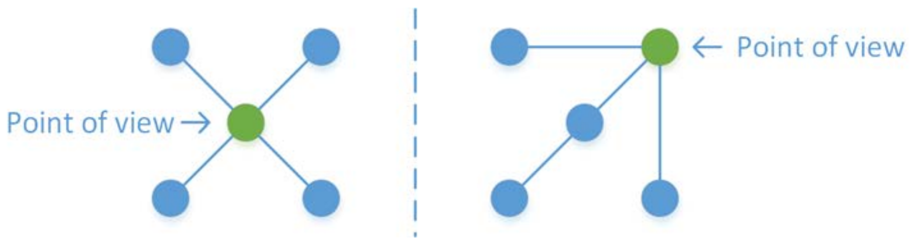

Figure 5-11 Shortest path seen from two different perspectives

Therefore, when the work node has found nodes among its neighbors with only one edge, it does a decision on their behalf not to relay. The nodes set as not relay, leaves the decision phase and do no longer respond to messages from this phase. The work node must decide if it should relay or not and it selects itself as relay unless it only has one edge. When the work node has made its decision, it sets a random timer for each neighbor with more than one edge. When the timer reaches 0, the work node passes the decision token on to the respective neighbor. The neighbor receiving the decision token start its own decision. This makes the algorithm spread in a ring from the trigger node.

The algorithm has been tested in MATLAB where it is verified to create a CDS for the network. To compare the performance with the centralized solution a simulation of the number of relays selected has been performed. From the results seen in Figure 5-12 it can be concluded the goal of stabilizing the number of relays selected was not achieved for the distributed algorithm. Figure 5-12 shows how the number of selected relays increases linearly for denser networks which is an undesirable behavior.

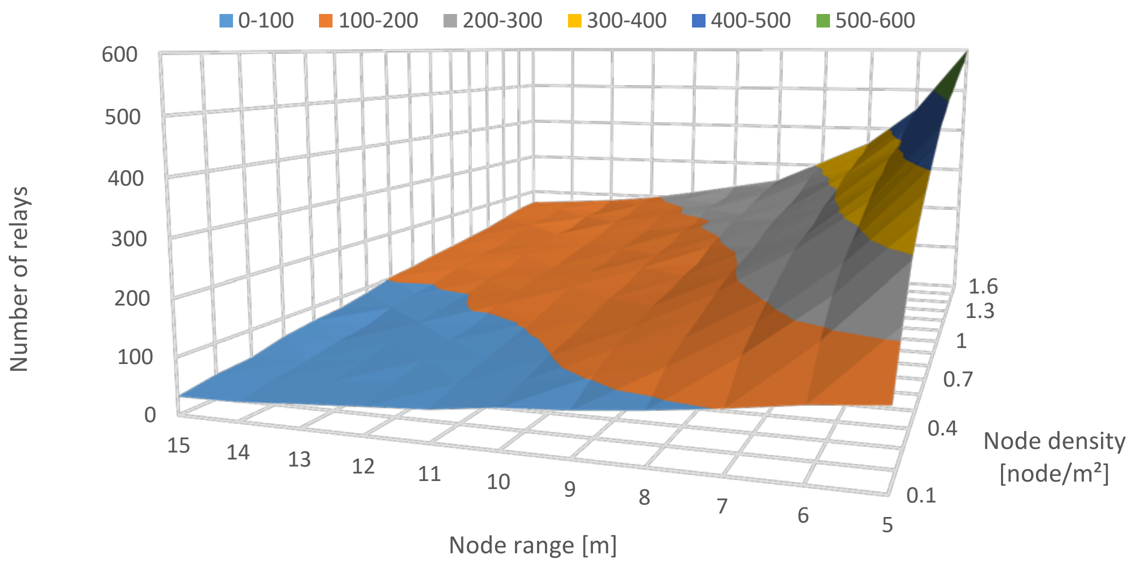

Figure 5-12 Distributed algorithm test for 50x50m area with varying node range and density

A cross section of Figure 5-13, where the node range is locked at 10m is displayed in Figure 5-13. It is clear the distributed algorithm has a linear increase. When the node density is increased with 1 node/m², the distributed algorithm selects an additional 107 relays.

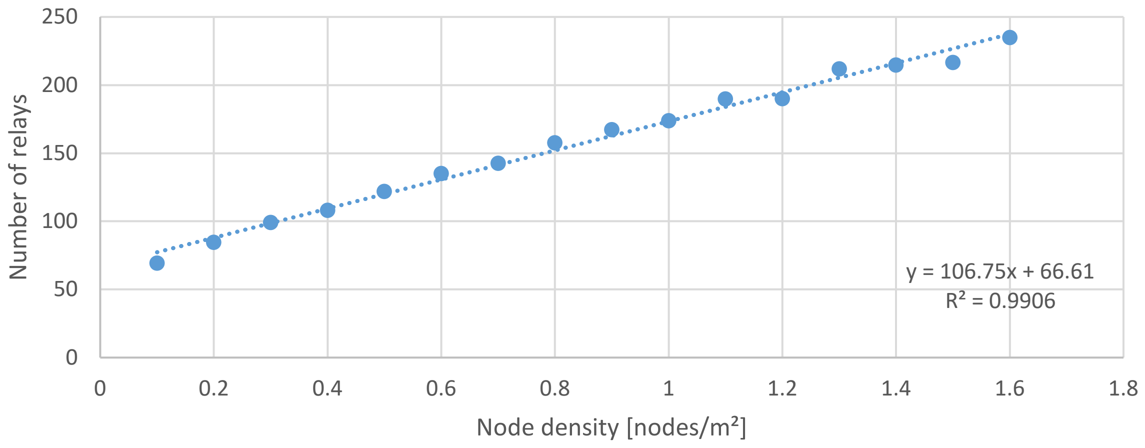

Figure 5-13 Number of nodes selected for different network densities

Furthermore, in Appendix B, an illustration that allows visual inspection of the centralized vs the distributed algorithm is presented. It is seen that the distributed algorithm preserves more cycles in the graph, were the centralized does not preserve any cycles. This is problematic, because it causes some messages to be relayed along an unnecessarily long path. To evaluate the impact of the broken cycles, simulations have been run on both the centralized and distributed graph. In the simulation, the average hop distance using the relays in the CDS is compared to the direct path. This has been tested on 10 different networks with 75 nodes, grid size 50 x 50 meter. The result is displayed in Figure 5-14. In the figure, it is seen that in the centralized solution, nodes must send packets an average of 1-2 hops more to reach its destination. Thus, even if the distributed algorithm choses more relays, it is a desired feature to have these cycles. Each time a packet must be relayed, the probability of collision increases.

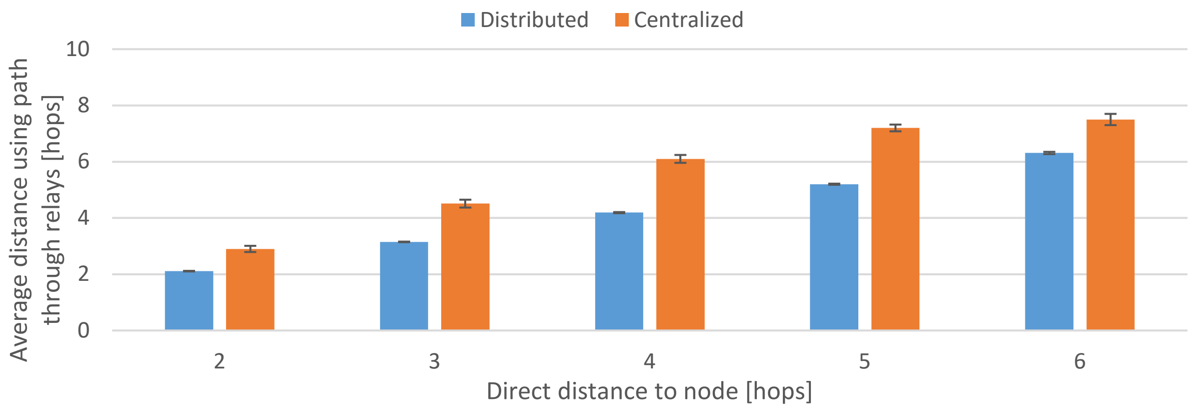

Figure 5-14 Difference between hop distance of centralized and distributed algorithms

The algorithm has now been developed to a stage, where it is both distributed and able to create a CDS solution. The algorithm scales with the density of the network, which is undesirable. On the other hand, the algorithm selects more relays than the centralized, which creates cycles in the network. Adding cycles to the network is preferred over having few relays, as this is thought to cause fewer collisions. The previous assumption that fewer relays equals fewer collisions is disregarded. Iteration four of the algorithm should select more relays than iteration two, but scale better with the density of the network than iteration three. Iteration four is left for future work, as the project must precede to the implementation to be able to thoroughly evaluate the algorithm within the timeframe.

6 | Implementation of algorithm in ns-3

This chapter describes how the designed algorithm is implemented. The algorithm is implemented as a state machine. This provides a skeleton for the algorithm with clear transitions between phases making implementation easier. Based on the state machine, the designed algorithm is implemented in ns-3 to enable more realistic evaluation compared to the ideal simulations done in MATLAB. The source code for the implemented algorithm is available at "Digital exam".

6.1 State machine

The state machine has three states, one for each phase in the designed algorithm. In the state machine, the trigger functionality of the network monitoring phase has been merged into the configuration initialization phase. Each state has events transitioning the state machine into a new state. There are two types of events:

- Internal event, is when something happens within a state of the state machine which causes a transition. A node deciding to be a relay is an internal event.
- External events, are events which the state machine has no control over that trigger at state change. A Provisioner asking a node to start the configuration or a node receiving a "do not relay" message, are considered as external events.

A diagram of the state machine can be seen in Figure 6-1. For simplicity, the diagram only shows events which cause transitions. Events like receiving information about the node connectivity, which do not change the states of the machine are not displayed in the diagram. The following sections discuss the states in the state machine.

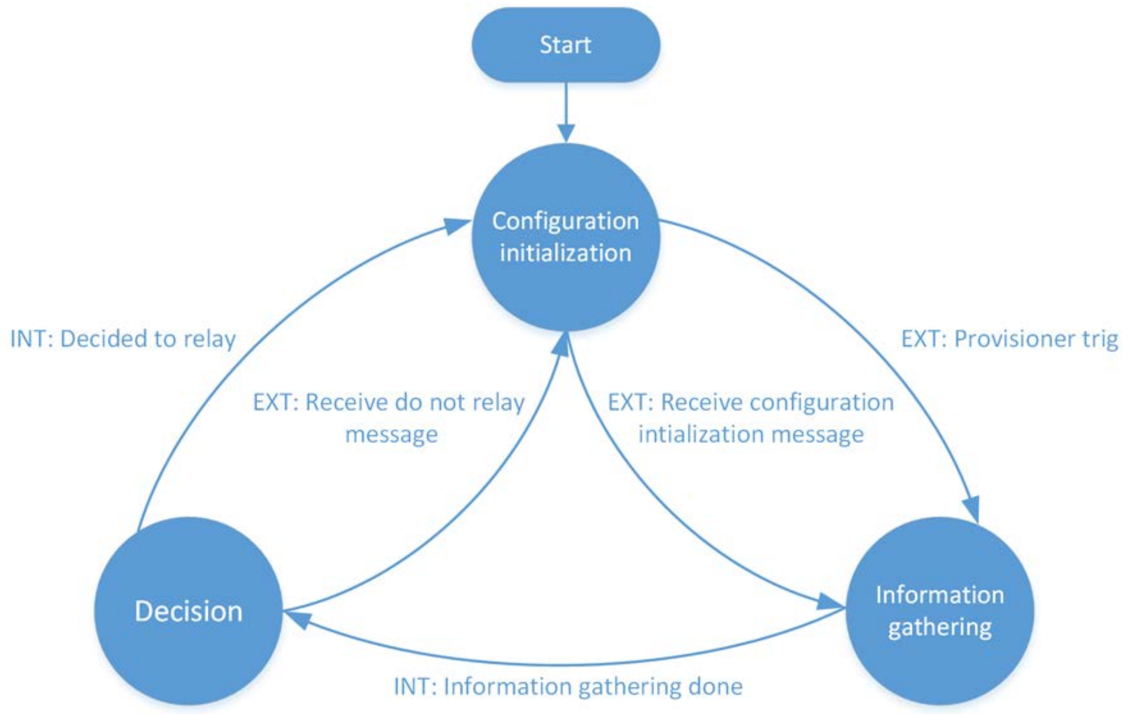
Start
Configuration initialization
INT: Decided to relay
EXT: Provisioner trig
EXT: Receive do not relay message
EXT: Receive configuration intialization message
Decision
Information gathering
INT: Information gathering done

Figure 6-1 Diagram of the state machine

6.2 Configuration initialization

In the implementation, the configuration initialization and network monitoring phases are combined, due to their limited functionality. A node starts in the network monitoring phase and stays here until it is asked to trigger configuration initialization by the Provisioner or the node receives a configuration initialization message. The messages used by the algorithm can be found in Appendix C. If the node is asked by the Provisioner to trigger it generates the configuration initialization message and broadcast it once. If the node receives the message from another node it must rebroadcast it once to make sure it reaches the whole network. If the message has already been received it is ignored. The message is rebroadcasted instead of relayed because the relaying feature is unlikely to be enabled in any node. That way the relaying of message is handled by the model layer and the algorithm can apply its own rules on how to repeat the message. It is believed, when a message is rebroadcasted by all nodes there is no need for the message to be broadcasted more than once, since a node likely would receive the message multiple times. The rebroadcasting happens with a random spacing to avoid collisions. When the message is broadcasted, the node waits for a short amount of time and proceeds to the information gathering phase. The waiting ensures the neighbor nodes had the time to resend the message and change state as well.

6.3 Information gathering

As described in the design of the information gathering phase in Section 5.2, the information gathering is done in two sub phases, discovery and information from neighbor. In the first sub phase the node broadcasts its "here I am" message three times. The messages used by the algorithm can be seen in Appendix C. The message is broadcasted with a random delay between the broadcasts. The node always listens for incoming "here I am" messages and if one is received it is saved into the network knowledge list. The BLE implementation in ns-3 uses a simple propagation model and therefore, the RSSI is constant between two nodes, as the value only depends on distance. If a more advanced propagation model is implemented and the RSSI starts to fluctuate, the value should be converted into a moving average or similar. This allows for a better representation of the RSSI. When all three "here I am" messages have been broadcasted, the node waits for some time to ensure all nodes in its neighborhood have finished broadcasting their "here I am" messages.

After the waiting time the node sends a reliable request to all discovered neighbors, one at a time, to request their view. This also happens with random spacing. When all neighbors have been requested, the node waits for some time and starts the decision phase. Since the message is reliable, there is no handling of cases when the message is not received due to the receiver being unavailable.

6.4 Decision

The distributed algorithm designed in Section 5.3.2 has been implemented in ns-3 as the decision phase of the state machine. The node waits in the begging of this phase until it receives the decision token or is told not to relay. If the node is told not to relay it goes to the configuration initialization phase. If it receives the decision token it will start the decision algorithm. The decision token is spawned by the node that generated the configuration initialization message.

The first step of the decision phase is to convert the information gathered into an adjacency matrix. The adjacency matrix contains all known nodes and the relay likelihoods as edges between the nodes. The collected information is also used to generate a neighbor list. It contains all the mesh addresses of the neighbors of the node. The node makes a shortest path from itself to all known nodes. The algorithm used for shortest path is Dijkstra's algorithm [16], since it is easy to implement. The algorithm runs for networks of E edges and V vertices in time $O(|V^2|)$. This is not the optimal solution since there are other shortest path algorithms which run in better times. One such algorithm is Thorup's algorithm [17] which runs in time $O(E + V \log\log V)$. Investigating the benefits of more complex shortest path algorithms, though, is not a focus of this project.

The shortest path is used to make the decision if the node should relay or not. The node checks how many nodes it is connected to. If it is connected to only one node, the node is an edge case and should not relay, else the node will decide to relay. The shortest path is also used to make two lists, one list of all the nodes with only one edge and another with nodes having more than one edge. The list of nodes having one edge consists of mesh addresses where the list of nodes with more than one edge consists of mesh addresses and relay likelihood. The list of nodes with one edge is now compared with the neighbor list and each node appearing in both lists are sent a "do not relay" message. The messages are sent as reliable unicast to ensure they are delivered. When all nodes in both lists have been informed, they should not relay, the list of nodes with more than one edge is sorted with respect to relay likelihood from highest to lowest. The node with the highest relay likelihood should be the best relay candidate. The nodes both in the list of nodes with more than one edge and the neighbor list are given a token. The node will do this by sending a "publish decision" message to the node with the highest relay likelihood, and after some time to the next one. This spacing should give the best relay candidate the chance to decide first, and thereby get a better result. When all nodes in the list have received a "publish decision" message the nodes go back to the configuration initialization phase.

The implemented algorithm is used to simulate results and do verification and evaluation of the designed algorithm.

7 | Results and discussion

This chapter presents the results from the designed and implemented relay selection algorithm, and discusses how well the algorithm performs. The results presented are a combination of data collected with MATLAB and ns-3. Further explanation of the methods and simulation parameters is presented before the results. The goal was to evaluate all performance indicators and verify the algorithm through simulation in ns-3. This, however, has not been possible, since limitations were found in the implementation of the BLE-mesh that could not be corrected in the timeframe of the project.

7.1 Limitation of BLE-mesh implementation

The BLE-mesh implementation in ns-3 has a few deficiencies, which unfortunately limits the extent of the evaluation.

- **Extended collisions:** the implementation of the BLE is done in a way that if a packet is under reception in one node, any other transmission of a packet will cause a collision. Thus, packet collisions are not depending on the signal strength of the colliding packet. Therefore, the collision probability increases rapidly when increasing the number of nodes.

- **Incoming segments**: a node can only receive one segmented messages at a time and corrupts the simulation if two nodes try to send a segmented message to it. This is not because of collisions, but because of errors in the logic, which handles segments and acknowledgments. This problem limits the amount of neighbors a node can have because if the node has many neighbors the probability of two nodes sending a reliable message to the same node increases. Additionally, more neighbors result in increased message sizes, which leads to more messages needing segmentation. Therefore, increasing the density of nodes would increase the probability of a corruption in the simulator. This makes it necessary to increase the random delays between messages in the information gathering phase and decision phase. These delays are larger than what is needed in a real-world situation, as this is a fix to get some data from the ns-3 simulator. Furthermore, the fix has not made it possible to evaluate the configuration time of the algorithm.

- **No signal of transmission success to model layer**: Whenever a reliable message is being transmitted, the lower transport layer retransmits the segments of the message up to three times. If the retransmissions fail, it gives up transmitting the message. However, the lower transport layer never signals to the model layer if this cancellation has happened. Thus, models are unaware that the message, which was wanted to be transmit reliably did not reach the destination.

7.2 Verification of implementation

Throughout the design phase, the different iterations of the algorithm have been verified based on varying node ranges and densities. This has been done in MATLAB, where the algorithms have been tested on networks with nodes placed uniformly within a 50x50 meters area. The results shown in Figure 5-5, Figure 5-9 and Figure 5-12 are based on at least 1000 different networks. The lowest node densities resulted in networks which were sparse, and the topology resembled a sparse tree. The highest densities used, were limited only by the time it took to run the verification. Node ranges were varied from 5 to 15 meters, which is considered around the boundary of the realistic range of Bluetooth class 2 devices that would adopt BLE-mesh.

Before a network was fed to the algorithm, it was verified that the network had a CDS solution. The networks had varying conditions such as node density and node placement. The networks were fed to the algorithm in MATLAB and the solutions calculated were verified to be a valid CDS solutions.

The same networks were used in verification of the ns-3 implementation of the algorithm, to verify both if the algorithm was implemented identically to the MATLAB implementation and to verify the solutions found. Due to the limitation in ns-3, it was only possible to run simulations with a maximum of 100 nodes. Even with a low density of nodes, some simulations resulted in solutions which were not valid, due to message segmentation errors. The incorrect solutions were caused by communication lockups caused by a buggy implementation of the lower transport layer. On other simulations, the algorithm generated a valid solution to the CDS relay selection problem. The implementation of the algorithm is deemed valid enough to do an evaluation of the performance.

7.3 Evaluation of solution provided by algorithm

A solution is a valid CDS generated by the implemented algorithm. The performance indicators of the solution to be evaluated are collisions, hop distance, latency and delivery probability, explained and described in Section 3.2. Collisions are directly extracted from ns-3 simulations. Hop distances are calculated in MATLAB based on ns-3 solutions. Latency and delivery probability are evaluated based on inference from the hop distance and collision simulation.

7.3.1 Collision

The collisions are evaluated in ns-3. The evaluation happens after the configuration has finished. When the network is configured, traffic is injected and the amount of collisions caused by this traffic is measured. Collisions were evaluated as a function of relays, node density and network load. Network load is the total number of packets generated by all the nodes in the network per second.

The traffic injected in the network is generated by a Poisson traffic model. This model is present on all nodes in the network. To have a large confidence in the collision data, there should be simulated on a variety of random networks, but time has not allowed for this. Therefore, all results present are only for one realization of a random network.

All the data extracted from the simulator are extracted from the BLE. It is in the BLE where collisions are registered, and it is also possible to see how many packets have been transmitted. Realistically, there is not a one to one relationship between number of transmissions and number of collisions. Transmission events are unique - one packet will only cause one transmission event. Collision events are not unique – they are logged by all nodes hearing the message. For this reason, duplicate collisions events are combined to one collision event in collection of result.

Solutions have been calculated based on random networks generated according to the specification listed in Table 7-1. The random network and the selection of relays in the network has been done by the MATLAB implementation of the relay selection algorithm. All the generated networks are checked to have a possible CDS solution before being imported into ns-3 for evaluation.

Network generation	
Area (length x width x height)	50m x 50m x 0m (2D)
Node placement	Uniformly distributed within the area
Message TTL	Uniformly distributed random number between 1 and 5

Table 7-1 Specification of the test networks generated for the simulations

The impact of the number of relays selected in a network is evaluated in terms of collisions. The CDS provided by the algorithm is compared to two general cases, all nodes relay and no relays. There is no CDS when no nodes relay, but the case is used as a baseline check of the collision probability in a network without relays. In these simulations, the node amount is fixed, and two network loads are simulated. The simulation parameters are specified in Table 7-2.

Parameters varied one by one in simulations	
Relay selection procedure	Algorithm, no relays and all relays
Network loads	10 and 100 packets/s
Parameters constant across all simulations	
Nodes (Density)	100 (0.04)
Simulation length	Approximately 10,000 packets transmitted
Nodes in network (nodes in CDS)	100 (37)

Table 7-2 Simulation parameters for evaluation of relays impact on collisions

The simulation results can be seen in Figure 7-1. It is expected that adding more relays to the network will cause more collisions as more relays increases the probability of two relaying at the same time.

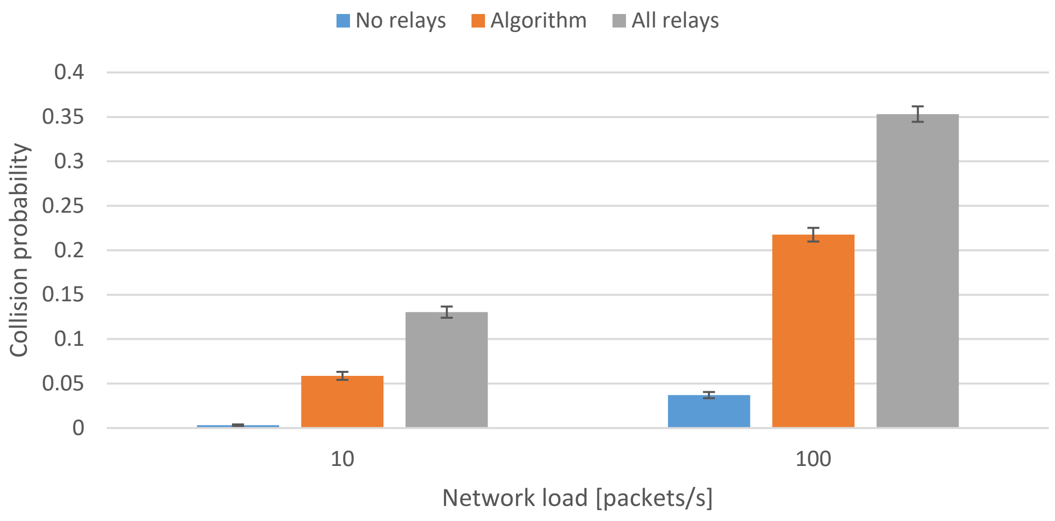

Figure 7-1 Comparison of collision probability between three methods of selecting algorithms

Intuitively, the CDS created by the algorithm should result in a collision probability somewhere in between the no relays and all relays case. Clearly, this is the case. The algorithms selection of relays leads to a collision probability closer to the all relay rather than the no relay case. This, however should be expected, because every packet generated in the network with no relays is transmitted once, while a packet generated in a network connected by relays is transmitted multiple times, increasing the collision probability. The higher the TTL, the more times a packet is relayed, thereby making it more probable for the packets to collide. Considering the no relay case as a baseline, it is apparent that when relays are introduced to the network, more collisions occur. Interestingly, the increase in collision probability caused by introducing relays is significant.

The number of relays introduced by the algorithm cause more than a quadrupling of the collision probability. Even though there is a clear penalty in terms of collision probability by introducing relays, it is a necessity if communication in the network should be possible. Increasing the network load causes more collisions. This is expected as the network becomes busier.

To further investigate the relation between network load and collisions, networks with varying densities and loads are simulated. Denser networks are expected to have more collisions as the algorithm selects more relays when the density increases. A high network load will cause more collisions as there are more packets traveling though the network at any given time. The specific simulation parameters are summarized in Table 7-3.

Parameters varied one by one in simulations	
Nodes (density)	50 (0.02), 100 (0.04) and 200 (0.08)
Network loads	0.1, 1, 10, 100, 1000 packets/s
Parameters constant across all simulations	
Simulation length	approximately 10,000 packets transmitted
Nodes in network (nodes in CDS)	50 (22), 100 (37), 200 (55)

Table 7-3 Simulation parameters for investigation of network load and density

The results of the simulations described in Table 7-3 are shown in Figure 7-2. It is apparent that the heaviest network load (1000 packets/s) results in the highest probability of collisions. However, it seems there is a lower bound on the collision probability. Somewhere between a network load of 1 and 0.1 packets/s the collision probability stabilizes on all three network densities.

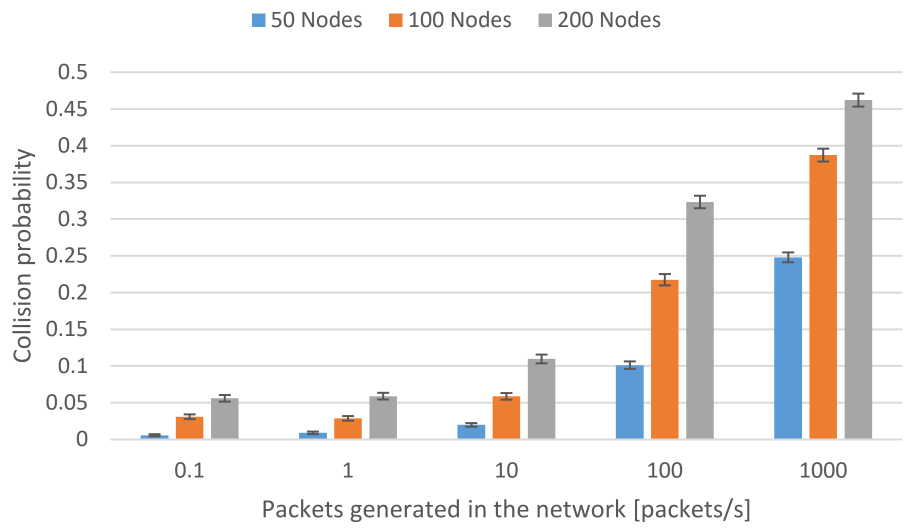

Figure 7-2 Collision probability as function of node density and network load

When the network load is low the probability of two simultaneous packet injections is low. Therefore, the majority of the collisions will be caused by the same packet being relayed by multiple relays. This could indicate that there is room for improvement in the way packets are being relayed. In the current implementation, a relay will add 0-100 ms random delay, before relaying. This delay could be increased. In Figure 7-2 there is a clear trend that increasing the number of nodes will increase the probability of collisions. Considering that, as the number of nodes increase, so does the number of relays, as seen in Figure 5-13, it is not surprising that the networks with more nodes result in a higher collision probability.

Injecting more traffic into the network, which is experiencing the highest collision probability gives a less significant increase in collision probability compared to injecting more traffic into the network with the lowest collision probability. Intuitively this is an expected result. More collisions in a network will have higher impact on the collision probability in networks with low collision probability.

To ensure that the collision probability is dependent on the number of relays selected, and not the number of nodes in the network, a final simulation was conducted. Here, two network loads were run on networks of 50, 100 and 200 nodes. The relays were selected by the centralized version of the relay selecting algorithm. Thus, the number of relays needed to cover the networks of 100 and 200 nodes were equivalent, as previously illustrated in Figure 5-8. As the results illustrated in Figure 7-3Figure show, the collision probability is not influenced by the number of nodes in the network, but by the number of relays in the network.

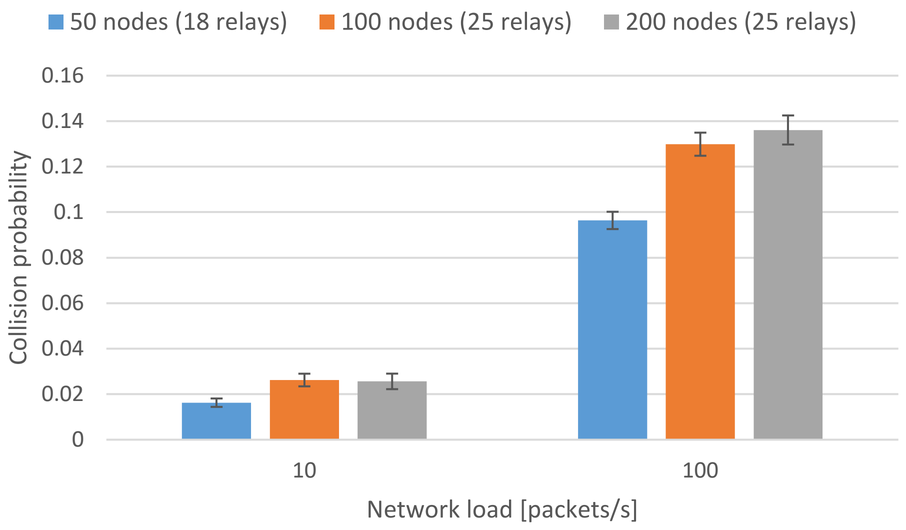

Figure 7-3 Collision probability measured on networks, where the relays have been selected by the iteration 2 of the algorithm (the centralized solution to relay selection)

From the evaluation of collisions in the network the following has been concluded. The number of relays in the network will as expected be a determining factor in the number of collisions, and higher network loads will increase the collision probability.

7.3.2 Hop distance, delivery probability and latency

Delivery probability and latency in a BLE-mesh network are both depended on how many hops a message must travel in the network. This is because each time a packet is relayed, it has a chance to collide. Each time a packet hops it adds a delay, which increases the latency. Because of that the delivery probability and latency are not evaluated further. Instead, the hop distance is evaluated by creating 10 different networks of 75 nodes, where both the ns-3 and MATLAB implementations create a CDS for the same network. The calculated CDS's are evaluated in MATLAB. The evaluation is a count of the distance between all nodes in the network. The results are an average on 10 networks and can be seen in Figure 7-4.

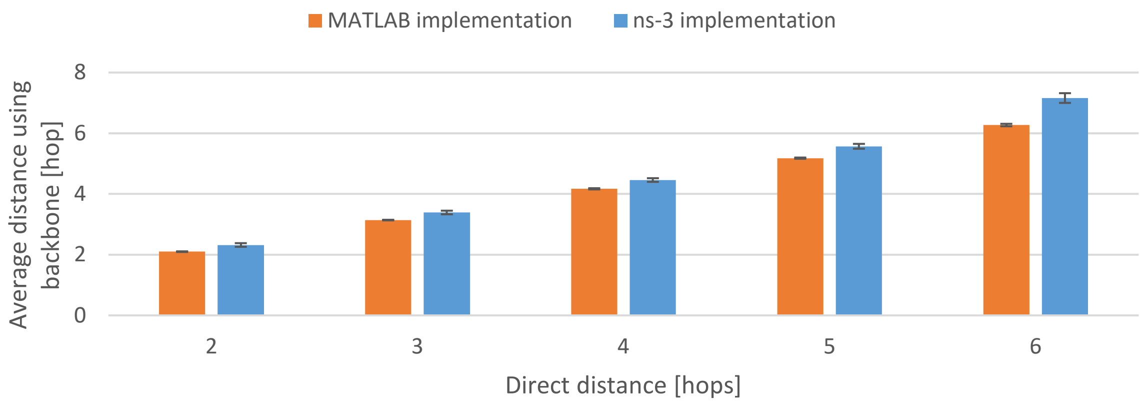

Figure 7-4 Hop distance for ns-3 and MATLAB implementation

The implementation in ns-3 preforms slightly worse than the MATLAB implementation. In most cases the two implementations select the same number of relays but not the same placement of the relays. The only difference in the logic of the two implementations is that the ns-3 implementation sorts the list of nodes with more than one edge, where MATLAB does not. This difference is believed to have caused the ns-3 implementation to select relays in a way that increases the hop distance. From Figure 7-4, it can also be seen that the selected relays almost always provide direct paths between nodes, which is a desired ability.

The CDS solution provided by the algorithm has now been evaluated. The next step in the evaluation is to evaluate the performance of the algorithm with respect to network load, computational complexity and memory consumption.

7.4 Evaluation of algorithm performance

In this section, the performance indicators regarding the algorithm itself are evaluated. Network load, computational complexity and memory consumption are evaluated based on analysis of the implementation of the algorithm. The configuration time is not evaluated, because of the delays included as work around to the limitation of the BLE-mesh implementation as described in Section 7.1.

7.4.1 Network load

Network load is the number of packets injected to the network by the algorithm. The number of packets injected to the network is different from the number of messages generated by the algorithm, because messages can be segmented into multiple packets. The network load is analyzed based on the different messages generated by the algorithm. The two message types, "Enter configuration mode" and "Here I am" are transmitted a fixed amount of times by each node. Generation of the other message types are dependent on the number of neighbors the generating node has. The exact number of messages generated per message type is presented in Table 7-4 and an explanation of the messages can be found in Appendix C.

Message	Number of messages generated by algorithm
Enter configuration mode	1
Here I am	3
Request full view	Number of neighbors
Reply full view	Number of neighbors
Do not relay + Publish decision	Number of neighbors

Table 7-4 Messages sent per node during configuration

The message type "Reply full view" is the only message type, which can be segmented into multiple packets. This is because the size of the message varies depending on the number of neighbors. The size of the message will be three bytes pr. neighbor. When the message is passed to BLE-mesh, the access layer prepends a one byte opcode to the message. The upper transport layer appends a 4 byte message integrity check. In the lower transport layer, any messages above 12 bytes are being segmented. The exact number of packets needed for the "Reply full view" message can be calculated by this expression:

$$\left\lceil \frac{3 \cdot NumberOfNeighbors + 5}{12} \right\rceil$$

The ceiling function is applied to the number, because segments, which are not full size will still require a packet. Calculating the total number of packets injected to the network by a node can be done with the following formula:

$$packets_{total} = 1 + 3 + 2 \cdot NumberOfNeighbors + \left\lceil \frac{3 \cdot NumberOfNeighbors + 5}{12} \right\rceil$$

A graph of the function can be seen in Figure 7-5. The number of packets send could be optimized by making the reply full view message into a broadcast because all neighbors need to receive this message.

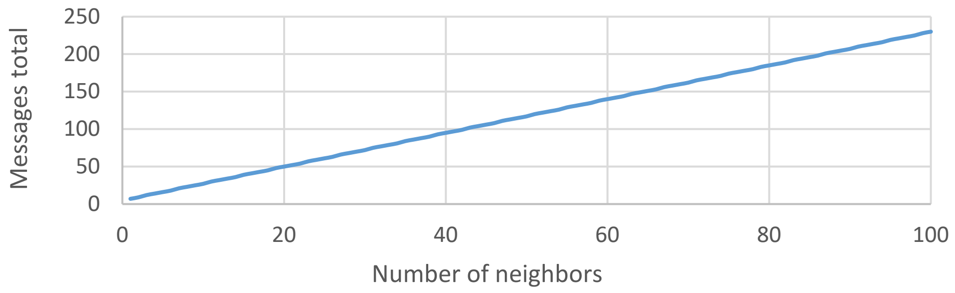

Figure 7-5 Number of messages send during configuration

7.4.2 Computational complexity

The complexity affects the time needed and power drawn in execution of the algorithm. The computational complexity is a function of the subgraph $G'(E',V')$ of the network which consists of the nodes and edges within the network visibility of a node. There are three major tasks within the algorithm that require computation. The major tasks are: creation of knowledge list, shortest path calculation and execution of relay assignment. All these task scale with the node density and network visibility. Creation of the knowledge list and execution of the relay assignment have not been investigated. As described in Section 6.4, the Dijkstra algorithm used for shortest path has complexity of $O(|V'|^2)$, which quickly grows to require a lot of computation. If the algorithm should be implemented on an actual BLE device, another better scaling shortest path algorithm should be used.

7.4.3 Memory consumption

The memory consumption is the amount of memory used by the algorithm. The precise memory consumption of the designed algorithm is unknown. But estimations on the largest memory objects used by the algorithm have been made. These objects are: the knowledge list, the adjacency matrix and the shortest path matrix. It is investigated how they scale when the density in the sub graph G' of the network increases, as more nodes means larger memory consumption.

The network knowledge list described in Section 6.3 consists of five bytes for each entry. The list has an entry for each edge visible to the node. Therefore, this list results in a memory consumption of 5 bytes times the number of edges, $|E'|$, in the subgraph. The adjacency matrix and shortest path matrix are both matrices with a size equal to the square of the number of visible nodes. Each entry in the matrices take up 1 byte of memory. The adjacency matrix is freed after the shortest path is created but during the creation they need to coexist. The two matrices together take up 2 bytes times the square of the number of vertices $|V'|^2$. Besides the dynamic memory consumption, the algorithm has a constant memory consumption. An estimation of the memory consumption of the algorithm is shown below.

$$Memory = \ 5bytes \cdot |E'| + \ 2bytes \cdot |V'|^2 + C$$

Where:

Memory	memory used by the major objects	Byte
E	set of undirected edges visible to the node	1
V	set of vertices visible to the node	1
C	constant memory consumption of the algorithm	Byte

The growth in memory consumption can be characterized by the O-notation: $O(|E'| + |V'|^2)$. To minimize the memory consumption the information that would go into the network knowledge list could be saved directly in the adjacency matrix. Additionally, the output of the shortest path calculation should not be an entire matrix, but a vector. This vector should hold, for each node in the graph, the previous node in the optimal path from the source. From this vector, the optimal path can be found by simply backtracking the route from destination to source

7.5 Discussion of algorithm

The verification and evaluation of the designed algorithm has shown the algorithm is able to provide configurations to random networks of different sizes. The success rate has been 100% in the MATLAB implementation but lower in the ns-3 implementation. Even though 100% was not reached in ns-3 the algorithm is deemed valid. It was observed the relay selection algorithm results in a collision probability, which is larger than the scenario without relays, but lower than the scenario where all nodes are relays.

The algorithm has one challenge which is scalability. Evaluation of performance indicators has shown scalability problems. There are two reasons behind the scalability problem, firstly the algorithm selects more relays when the density is increasing, this can be seen in Figure 5-13.

For an ideal algorithm, the number of relays should stabilize for a constant area. The behavior of selecting more relays in denser network can be seen in the collision results where the network with 200 nodes has more collisions than the network with 50 nodes even though the network load is the same. In Figure 7-3 it was apparent, that the number of relays had the most significant influence on the collision probability. Still, minimization of the number of relays should be done with care. At some point, even though it is possible to remove relays from the CDS, this would result in worse performance of the network. The reason being, that removing more relays breaks cycles in the network. This results in longer hop distances between nodes, which decreases delivery probability. The algorithm also scales in memory consumption, computational complexity and network load generated by the algorithm as a function of density. This combination leads to that the current implementation of the algorithm has a maximum density it can handle. When this maximum is reached, the network will either brake because of too many relays in the network or because the nodes do not have the resources to complete the configuration. The maximum has not been found do to limitation on the simulator as stated in Section 7.1.

8 | Conclusion

The goal of this project was to design a self-configuration algorithm for Bluetooth low energy (BLE) mesh networks. BLE-mesh networks introduce some limitations and constraints for the design of the algorithm. BLE-mesh devices are designed to be inexpensive and deployed in large numbers. The BLE-mesh networks have been simulated in a free space environment where placement of the nodes was uniformly distributed. The performance indicators identified as necessary to evaluate the performance of the algorithm can be split into two categories. The first category evaluates resource consumption of the algorithm by analyzing configuration time, network load, memory consumption and computational complexity. The second category evaluates the solution produced by the algorithm in terms of collisions, packet delivery, latency and hop distance in the network. Furthermore, the scalability of the algorithm is addressed.

Three iterations of the algorithm, creating a connected dominating set (CDS), were designed. The last iteration takes a distributed approach to the relay selection problem. The algorithm provides a CDS, but the size of the CDS scaled with the density of the network, which is an undesirable behavior. Although the algorithm does not provide a minimum CDS (MCDS), this was not deemed as a considerable drawback, since the MCDS does not always provide the best possible route for the messages within the network. Based on the analytical evaluation of the performance of the algorithm it was noted that the algorithm has problems with scalability.

The evaluation of the algorithm was performed in simulation environment where the BLE-mesh behavior was implemented. Only the core functionality of the BLE-mesh was implemented in the test environment. From the simulations, denser network will result in higher collision probability since more nodes will be selected as relays. Higher network loads will also result in increasing the collision probability. Some collisions are caused by packets being relayed. Therefore, at low network loads the collision probability stabilizes.

The algorithm designed in this project can autonomously configure BLE-mesh networks in a way which ensures full connectivity. The network conditions provided by the algorithm exacerbates as a function of density.

9 | Future work

This chapter describes several topics which could have been examined or further improved to enhance the performance of the algorithm.

The next iteration of the algorithm should address the scalability issue. The algorithm should be improved in a way which ensures the number of selected relays remains constant for the same area of the network regardless of the node density. One possible way to address this issue is to start using the choices from the former nodes to do a better decision. Moreover, the information gathering could be improved. The current information gathering informs all the neighbors about the network knowledge using unicast messages. This could be improved by broadcasting to all neighbors resulting in reduced number of messages.

The simulation environment could be improved by removing the limitations in the BLE-mesh implementation. It could also be improved by adding features such as: an error model introducing bit errors, adding error check to the BLE-mesh and upgrading the pathloss model to better reflect the real world. Furthermore, the simulation could be moved into a more realistic scenario, such as a building.

The conceptual design of the algorithm has five phases, but only three have been implemented. Future development could address and implement the missing phases. The network monitor phase should be designed to incorporate self-healing in the network, giving it the ability to auto detect typology changes, and act upon these changes. Adding the relay intelligence phase allows the algorithm to optimize the solution after it has been generated. The relays can also start improving their routing capabilities, by monitoring the traffic being relayed.

Bibliography

[1] L. Columbus, "forbes," Forbes Media LLC, 27 November 2016. [Online]. Available: https://www.forbes.com/sites/louiscolumbus/2016/11/27/roundup-of-internet-of-things-forecasts-and-market-estimates-2016/#7d549bf6292d. [Accessed 20 May 2017].

[2] Postscapes, "postscapes," [Online]. Available: https://www.postscapes.com/what-exactly-is-the-internet-of-things-infographic/.

[3] J. Kastrenakes, "The Verge," [Online]. Available: https://www.theverge.com/2015/1/5/7497537/samsung-iot-internet-of-things-vision-presented-at-ces-2015-keynote.

[4] "Postscapes," 14 November 2015. [Online]. Available: http://www.postscapes.com/internet-of-things-definition/. [Accessed 20 Februar 2017].

[5] A. Nordrum, "Spectrum (IEEE)," 18 August 216. [Online]. Available: http://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated. [Accessed 21 February 2017].

[6] Gartner, "Gartner," 10 November 2015. [Online]. Available: http://www.gartner.com/newsroom/id/3165317. [Accessed 21 February 2017].

[7] B. S. p. a. confidential, "Mesh, revison: d09219," 10 jan 2017.

[8] B. S. Mesh Working Group, Mesh, Bluetooth® Specification, Bluetooth SIG, 2017.

[9] J. W. Fei Dai, "An Extended Localized Algorithm for Connected Dominating Set Formation in Ad Hoc Wireless Networks," 2004.

[10] M. T. N. S. Akshaye Dhawan, "A Distributed Greedy Algorithm for Constructing Connected Dominating Sets in WSN," *Applications and Innovations in Mobile Computing (AIMoC), 2014,* 2014.

[11] Y. Wu, F. Wang, M. Thai and Y. Li, "Constructing k-Connected m-Dominating Sets in Wireless Sensor Networks," *http://ieeexplore.ieee.org,* 31 10 2007.

[12] R. W. Thomas Moscibroda, "Maximizing the Lifetime of Dominating Sets," 2005.

[13] Y. Kukimoto, "University of Colorado VLSI group," 1996. [Online]. Available: http://vlsi.colorado.edu/~vis/doc/VisUser/vis_user/node4.html.

[14] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Verification_and_validation.

[15] Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Path_loss.

[16] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik,* vol. 1, pp. 269-271, 1959.

[17] M. Thorup, "Integer priority queues with decrease key in constant time and the single source shortest paths problem," *Journal of Computer and System Sciences,* vol. 69, pp. 330-353, 2004.

[18] R. M. Karp, "Reducibility among Combinatorial Problems," in *Complexity of Computer Computations*, Springer, 1972, pp. 85-103.

[19] K.-H. Kim and K. G. Shin, "Self-Reconfigurable Wireless Mesh Network," *IEEE/ACM TRANSACTIONS ON NETWORKING,* no. 2, pp. 393-404, April 2011.

[20] H. Silva, R. Holanda, A. Santos and M. Nogueira, "A framework for self-configuration on WMNs aware of performance and security issues," *2010 International Conference on Network and Service Management,* 2010.

Appendix

A Visualization of iteration 1 & 2 of the algorithm

This Appendix shows visualizations of the centralized algorithm (iteration 1 & 2) of what happens when varying the numbers of relays and node range in a static grid. If a node is green it has been selected as a relay and if red, it has not been selected to be a relay.

Vary number of nodes:

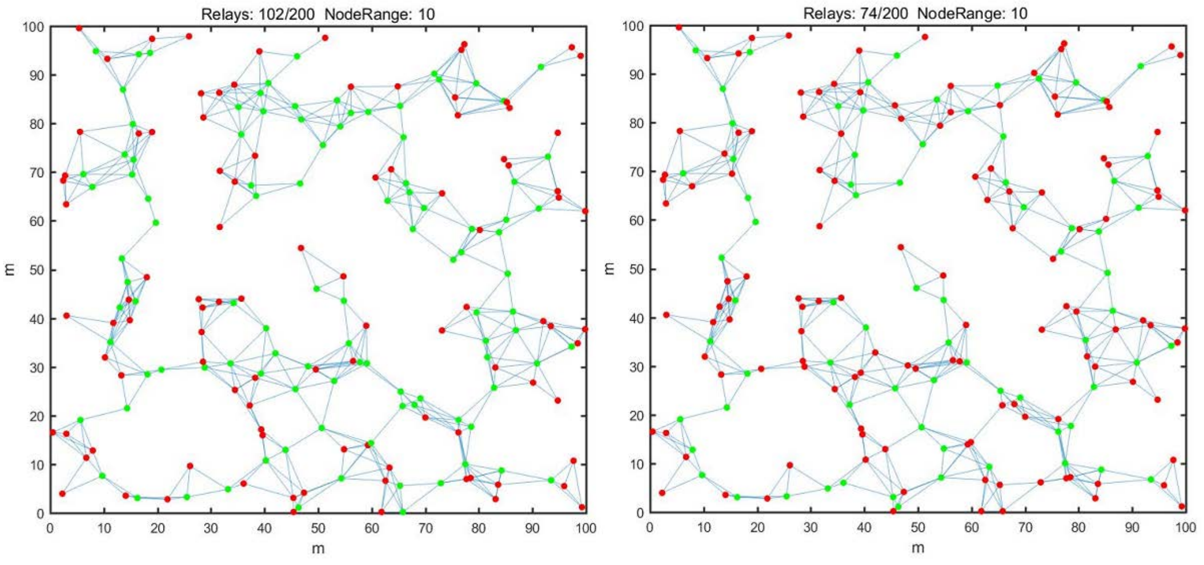

Figure A-1 To the left is our algorithm iteration 1 and to the right iteration 2

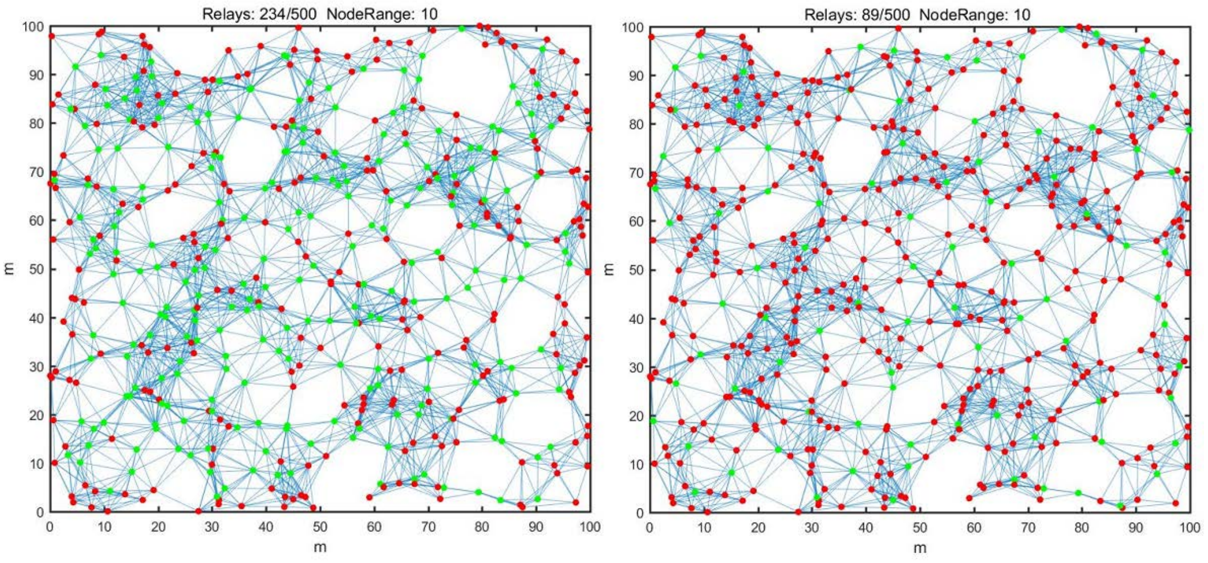

Figure A-2 To the left is our algorithm iteration 1 and to the right iteration 2

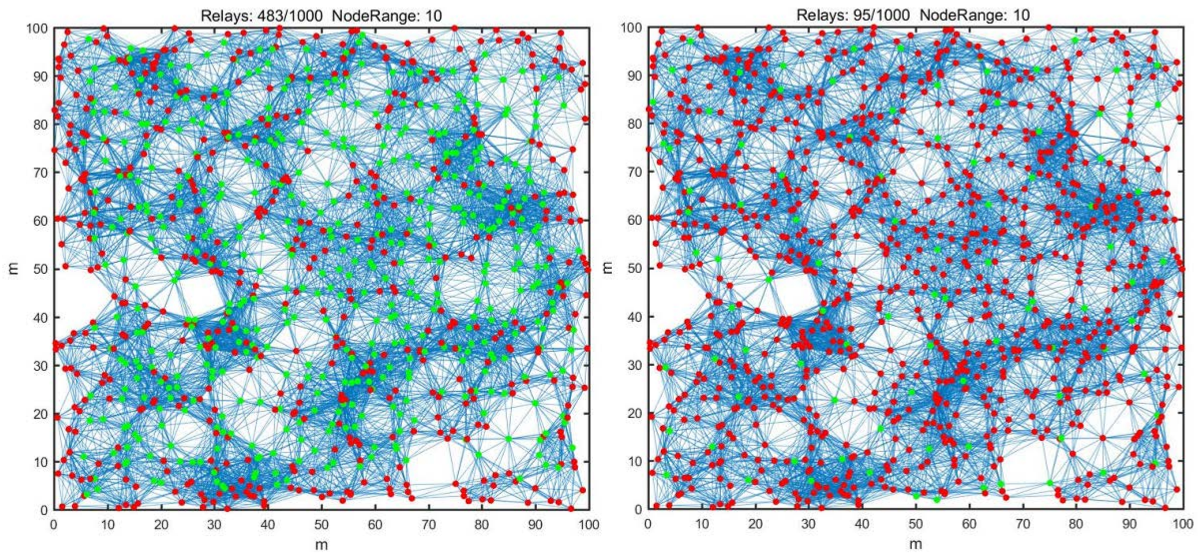

Figure A-3 To the left is our algorithm iteration 1 and to the right iteration 2

By varying the number of nodes, it is seen the algorithm without logic is selecting too many relays in the densest areas compared to the algorithm with logic.

Vary node Range:

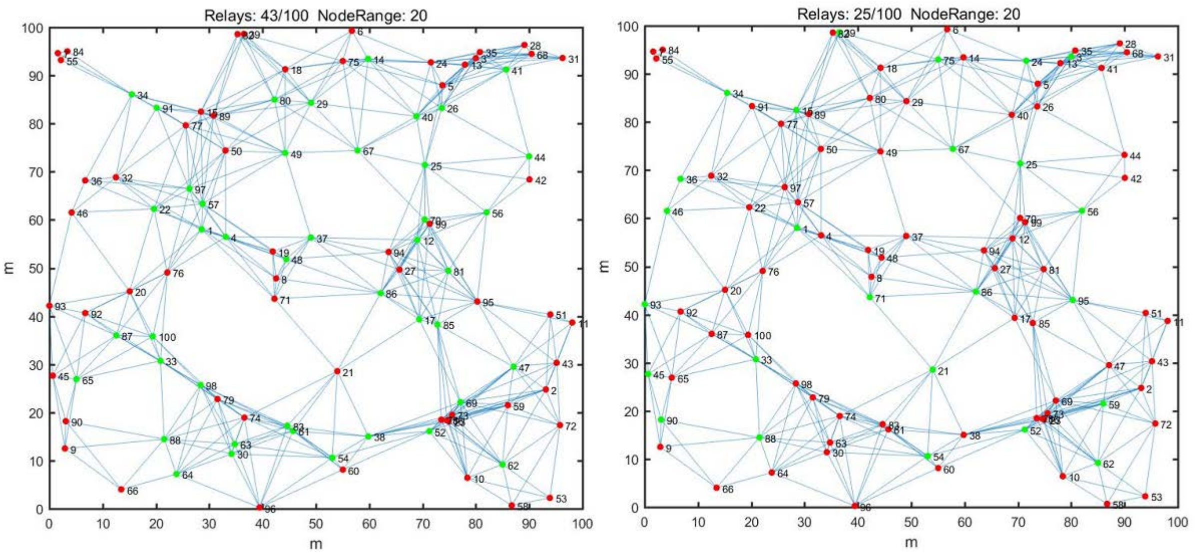

Figure A-4 To the left is our algorithm iteration 1 and to the right iteration 2

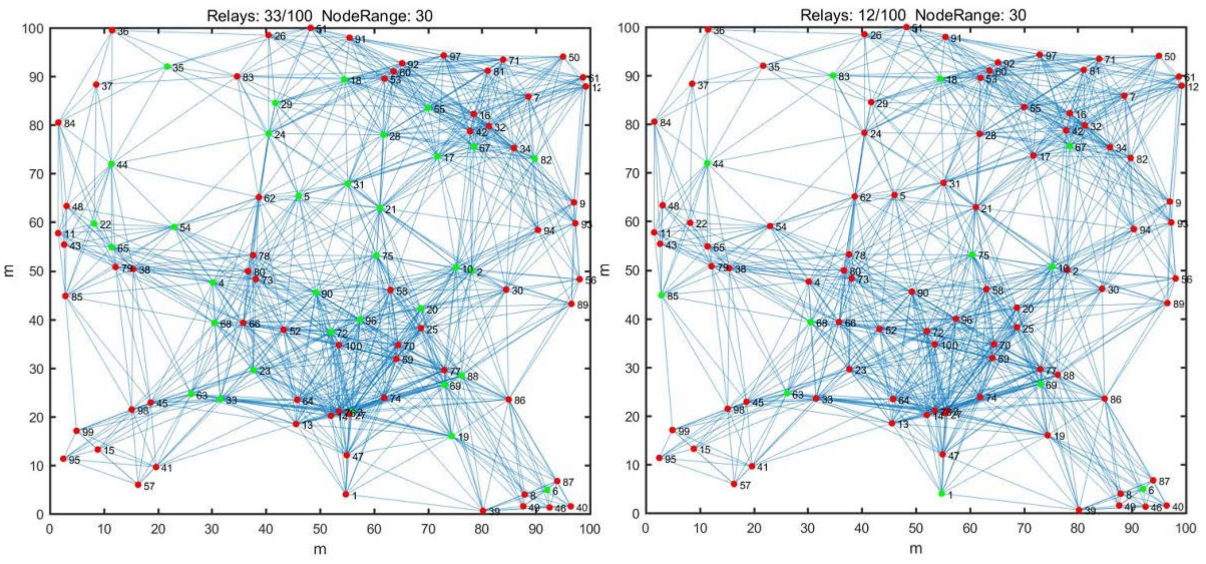

Figure A-5 To the left is our algorithm iteration 1 and to the right iteration 2

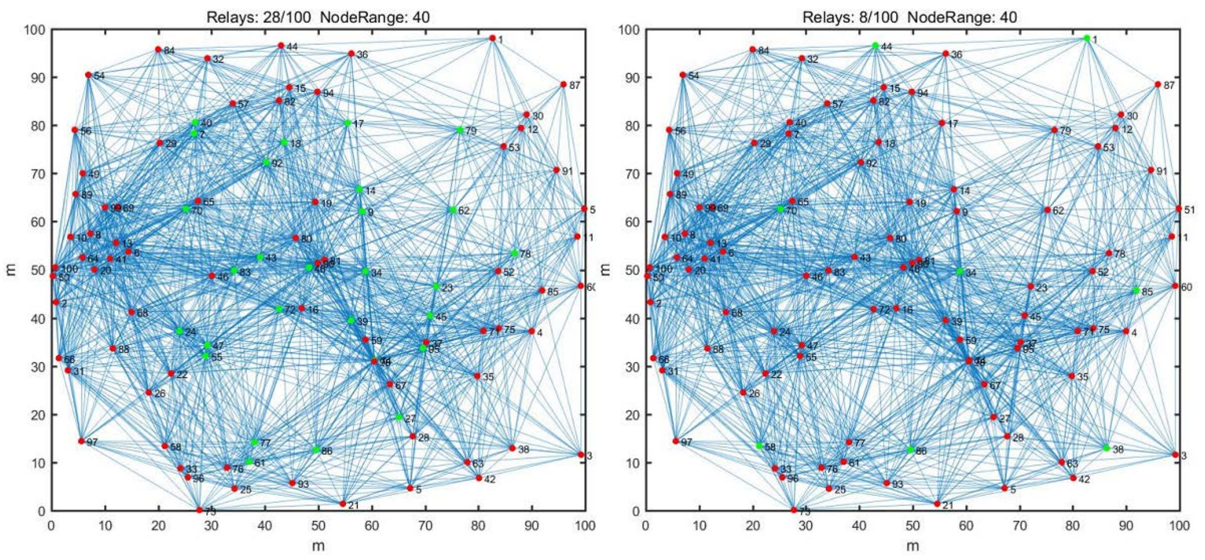

Figure A-6 To the left is our algorithm iteration 1 and to the right iteration 2

Varying the range of the nodes “can be seen” as making the grid size smaller, and you can say the density from a node perspective is increasing. This leads to the same conclusion, when more nodes are added, the algorithm without logic has a hard time selecting relays in the densest areas.

B Hop distance in iteration 2 & 3

This Appendix shows visualizations of how the centralized (iteration 2) versus the distributed algorithm (iteration 3) have selected their CDS. If a node is green it has been selected as a relay and if red, it has not been selected to be a relay.

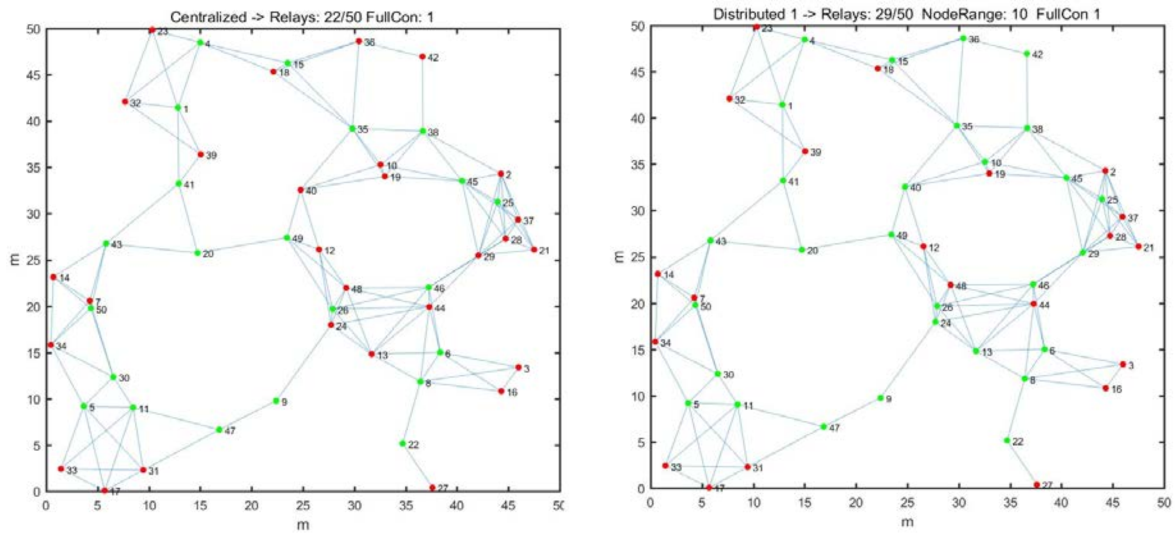
Centralized -> Relays: 22/50 FullCon: 1
Distributed 1 -> Relays: 29/50 NodeRange: 10 FullCon 1
m
m

Figure B-7 Centralized vs distributed relay selection

The simulations have been with 50 nodes, range 10 meter and grid size 50 x 50 meters. From Figure B-7 it is clear that the centralized breaks all cycle in the graph, where the distributed maintains cycles. By breaking all cycles, the centralized selects fewer relays at the cost of a bigger hop distance between some nodes. This is not desirable as some nodes in the centralized will get high latency and packet loss probability because they must send the message on a detour to reach the destination. An example of this is if node 35 wants to send a message to node 49. In the distributed algorithm, the distance is two hops where in the centralized it is six hops.

C Algorithm messages

The self-configuration algorithm requires several special messages to move the needed information around. This Appendix describes the different messages needed for the algorithm and how they are formatted. The messages could be designed in two ways. One where an opcode is used for all the messages and then the payload contains a sub opcode. Alternatively, each message could have its own opcode. We chose the solution where each message has its own opcode. This will give a more efficient data transfer as you can have larger data amounts in your payload when you do not need the sub opcode. The chosen starting opcode is randomly selected. The opcodes are one byte long. The mesh specification allows for opcodes up to 3 bytes long. The messages are presented in the table below.

Messages	**Opcode**	**Payload**	**Comment**
Enter configuration mode	51	Trigger number (1 byte)	The trigger number should be initialized at 1 and incremented every time the node triggers a self-configuration. This number wraps around at overflow. The messages must be send to the broadcast address with a TTL of 0
Here I am	52	movement probability (1 byte)	The messages must be send to the broadcast address with a TTL of 0
Request full view	53	None	Send as a request at a single node. This is send as reliable message.
Reply full view	54	N chunks of 3 bytes, where each chunk has the following structure: 2 byte unicast addressee of the node and one byte relay likelihood. Where N is the number of neighbors.	The node must send all entries of known neighbors.
Do not relay	55	None	Send as reliable unicast message
Publish decision	56	None	Sent as reliable unicast to maybe nodes.

D Minimum vertex cover optimization

This Appendix investigates the minimum vertex cover problem. Furthermore, it is investigated how much it would require solving it as an optimization problem. This will give us an insight in the computational load of an algorithm where optimization is performed.

To work with the network in a mathematical sense it has been converted into a graph theory problem. The network is described as a graph $G(V,E)$, where the vertices V is a set consisting of all the nodes in the network and edges E is a set containing all connections between the nodes. The weight of the edges is the relay likelihood found in the information gathering phase. The problem of finding the minimum number of relays can now be formulated as finding a minimum subset V' of the set V which ensures that the set V' can be connected to every node in V with one edge.

This problem is a general problem in graph theory called minimum vertex cover. The problem is to find a minimum subset V' of V where every edge in E has at least one endpoint in the set V'. The vertex cover problem is a known NP-complete problem [18], which means that this problem does not have a deterministic solution in polynomial time. This implies that a found minimum might not be a global minimum. The problem can be formulated as:

$minimize \sum_{v \in V} x_v$	Minimize the total number of nodes
Subject to:	
$x_u + x_v \geq 1 \; for\ all \; \{u,v\} \in E$	Cover every edge in the Graph
$x_v \in \{0,1\} \quad for\ all\ v \in V$	x_v is an indicator variable, which is 1 if the vertex v is within the vertex cover set, and 0 if v is not in the vertex cover set

The minimum vertex cover does not provide the desired solution because it chooses more relays than needed. This is due to how the problem is constraint, where every edge must be covered with at least one node from the set. This is illustrated in Figure D-8, where a network of four nodes has been used to test the minimum vertex cover problem. Node 1 and 3 have been selected as relays and thereby marked with a green circle. From a network perspective, it would have been enough only to choose one of them, as they both connect all nodes in the network. The problem in this example is by only selecting 1 node, 1 or 3, two edges are not covered and the constraint, $x_u + x_v \geq 1 \; for\ all \; \{u,v\} \in E$, would not be fulfilled.

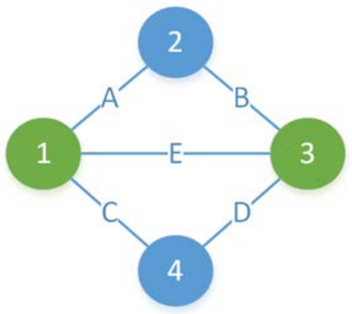

Figure D-8 Example of minimum vertex cover

The minimum vertex cover problem is of the type integer linear programming (ILP), which is a class of optimization problems that allows nonlinear integer constraints. ILP with binary constraints is also a NP-complete problem [18]. A simulation was created to verify performance of the minimum vertex cover. It is simulating the relay selection for a random network in a constant area. The area is a 100 x 100 meter free space area. The nodes are uniformly distributed in the area. The simulation test shows how the relay selection develops as function of node range and node density.

It is expected that the relay percentage is high for all scenarios in the simulation, due to the constrain that every edge must be covered, as seen in Figure D-8. It is also expected that the relay percentage increases with node density or node range as there are more edges introduced into the graph. If this would have been a valid solution, the percentage of relays would have stabilized as node density or node range is increased, which is the exact opposite of what the simulation shows. The output of the minimum vertex cover problem simulation can be seen in Figure D-9.

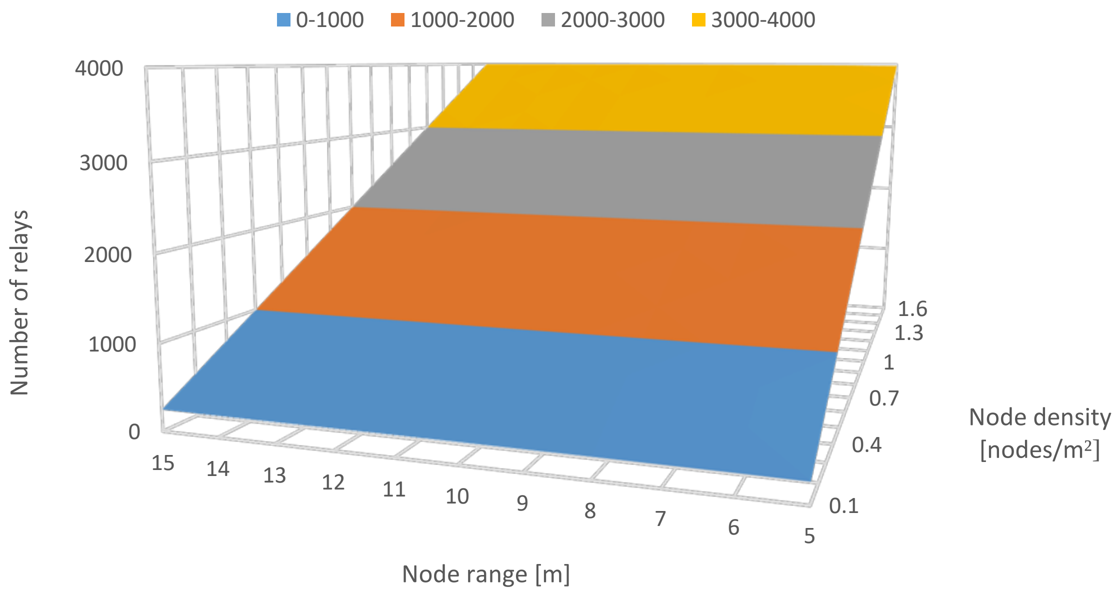

Figure D-9 Minimum vertex cover simulation

From the outcome of the simulation it has been decided not to use this approach. Additionally, the optimization problem is computationally heavy, which means it would not be realistic to implement it on anything less powerful than a notebook computer. Furthermore, to address the poor scaling in the number of selected relays more constraints should be added to the problem. This would only increase the computational complexity. Therefore, the relay selection algorithm is not designed as an optimization problem. Instead, it is designed in logic based on a simple graph collected in the information gathering phase.